User Manual

Skkynet VINE™

Version 7.3

Skkynet Cloud Systems, Inc.

June 03, 2014

Skkynet VINE™: Version 7.3

Versatile integration for networking Excel.

Published June 03, 2014
Skkynet Cloud Systems, Inc.

2233 Argentia Road
Mississauga, Ontario L5N 2X7
Canada

Toll Free: 1 888.628.2028
Fax: 1 888.705.5366

Information Email: info@skkynet.com
Web Site: www.skkynet.com

Copyright © Skkynet Cloud Systems, Inc.

Revision History

Revision 7.3-1 May 2014
Initial release of documentation.

Copyright and trademark information.

Copyright Notice

Copyright © Skkynet Cloud Systems, Inc. and/or its subsidiaries.
Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written consent of Skkynet Cloud Systems, Inc. Skkynet Cloud Systems, Inc. assumes no responsibility for any errors or omissions, nor do we assume liability for damages resulting from the use of the information contained in this document.

Trademark Notice

VINE™, Skkynet™ and their logos are trademarks of Skkynet Cloud Systems, Inc. DataHub®, Cogent™, Cascade™, QuickTrend™, WebView™, OPC Tunneller™, OPC Logger™, OPC Bridge™, System Monitor™, Gamma™, Historian™, TextLogger™, CIF Driver™, DataSim™, DataPid™ are trademarks of Real Innovations International LLC and used under license. All other trademarks are the property of their respective owners.

Table of Contents

1. Installation **1**

1.1. System Requirements and Installation 1
1.2. Installing Licenses 1
1.3. Installing VINE as a Service 2
1.4. Performing a Silent (Unattended) Install 7
1.5. Configuration Files 7

2. Getting Started **11**

2.1. Running Skkynet VINE 11
2.2. Test with simulated data 14
2.3. Test with Excel 15
2.4. Connect to remote data 16

3. Excel Connections **19**

3.1. Getting Data into Excel 19
3.1.1. Method 1 - Drag and Drop using **DDEAdvise** 19
3.1.2. Method 2 - Excel Macros using **DDERequest** 21
3.1.2.1. Create a macro ..... **??**
3.1.2.2. Add a Control Button 21
3.1.2.3. Receive the data ..... **??**
3.2. Getting Data out of Excel 22
3.2.1. Method 1 - Configuring **DDEAdvise** loops in Skkynet VINE 23
3.2.2. Method 2 - Writing Excel macros that use the **DDEPoke** command 25
3.2.2.1. Create a macro ..... **??**
3.2.2.2. Add a Control Button 26
3.2.2.3. Send the data ..... **??**
3.2.2.4. Additional Pointers 26
3.3. Working with Ranges 27
3.3.1. Getting a Range out of Excel 27
3.3.2. Getting a Range into Excel 28
3.3.3. Sample Excel Macros for Arrays 31

4. Tunnel/Mirror for Networking **33**

4.1. Introduction 33
4.2. Configuring the tunnelling master 33
4.3. Configuring a tunnelling slave 34

5. Bridging **38**

5.1. Introduction 38
5.2. Configuring Bridges 38
5.2.1. Point-to-point configuration 38
5.2.2. Making transformations 40
5.3. Creating New Points 42

6. Using QuickTrend **44**

7. Write to a Database **48**

7.1. Introduction 48
7.2. Quick Start 48
7.3. Configuring the Queue, Store and Forward 52
7.4. Setting up the DSN (Data Source Name) 54
7.5. Configuring a Database Table 56
7.6. Key Columns 60

7.7. Assigning a Trigger 61
7.8. Setting Trigger Conditions 63
7.9. Configured Actions 68
8. Query a Database 69
8.1. Introduction 69
8.2. Quick Start 69
8.3. Setting up the DSN (Data Source Name) 72
8.4. Configuring a Database Query 74
8.5. Assigning a Trigger 77
8.6. Setting Trigger Conditions 79
8.7. Configured Actions 82
9. System Monitor 84
9.1. Introduction 84
9.2. Configuring the System Monitor 84
9.3. Monitoring Systems Across a Network 87
10. Email and SMS 91
10.1. Introduction 91
10.1.1. How it works 91
10.1.2. A note about SMS text messages 91
10.2. Configuring the Mail Server 92
10.3. Sending a Test Message 93
10.4. Defining the Email Message 97
10.5. Assigning a Trigger 99
10.6. Setting Trigger Conditions 101
10.7. Configured Actions 104
10.8. HTML Message Examples 104
10.8.1. An HTML Message with Embedded Data Points 104
10.8.2. An HTML Message with a Table Created in Code 106
10.9. Dynamically Changing Email Subjects and Recipients 108
11. VINE Scripting 109
11.1. Tools 109
11.2. VINE ODBC (Open Database Connectivity) Scripting 109
11.3. VINE Windows Scripting 110
12. Security 112
12.1. How to Configure ..... **??**
12.2. SSL and Firewalls 114
12.3. User Authentication 114
12.4. Authorization and User Permissions 116
12.5. Permissions for the VINE Command Set 119
12.6. Passwords 122
13. Working With Data 123
13.1. Data Points 123
13.1.1. Creating New Points ..... **??**
13.1.2. Deleting Points ..... **??**
13.1.3. Viewing Data Points 123
13.1.4. Point Size Limits 123
13.2. Data Communication Concepts 124
13.2.1. Send and Receive Data 124
13.2.2. Client - Server 124

13.2.3. Synchronous and Asynchronous Communication....124
13.3. Data Exchange Protocols....125
13.3.1. DDE Protocol....125
13.3.2. TCP and Tunnelling/Mirroring....125
13.3.3. The VINE API....125
13.4. Data Organization....126
13.4.1. Data Domains....126
13.4.2. Assemblies, Subassemblies, Attributes, and Properties....126
13.4.3. Attributes and Types....127
13.4.4. Example 1: Attributes and Types....127
13.4.5. Example 2: Private Attributes....129
14. Optimizing Data Throughput....131
14.1. Binary Mode Tunnel/Mirror (TCP) Connections....131
14.2. Tunnel/Mirror (TCP) Connections for Slow Networks....131
14.3. Old Value Queuing....131
14.4. Un-Buffered Delivery....132
14.5. Screen Output....132
14.6. CPU Saturation....133
14.7. How to Optimize....133
14.7.1. Tunnel/Mirror (TCP) connections....133
14.7.2. DataHub C++ API....135
14.7.3. Gamma scripts....135
15. Properties Window....137
15.1. General....137
15.2. Tunnel/Mirror....138
15.3. Bridging....142
15.4. DDE....146
15.5. Quick Trend....148
15.6. Database....152
15.7. Historian....156
15.8. System Monitor....160
15.9. Email/SMS....162
15.10. Redundancy....164
15.11. Update....168
15.12. Scripting....169
15.13. Security....171
15.14. Licenses....174
16. Other Windows and Programs....178
16.1. Data Browser....178
16.2. Connection Viewer....179
16.3. Event Log....181
16.4. Script Editor....182
16.5. Script Log....182
16.6. DataSim - a data simulation program....183
16.7. DataPid - a PID loop data simulation program....185
16.8. Service Manager....188
17. Using VINE Commands....194
17.1. Command Syntax....194
17.2. Return Syntax....194
17.3. Sending Commands by TCP....195

18. Troubleshooting **197**
A. DDE Overview **200**
B. Excel Macro Library **203**
B.1. Configure Excel to receive data from Skkynet VINE (using DDEAdvise) 203
B.2. Write data from Excel - User initiated (using DDEPoke) 204
B.3. Write data from Excel - Automatically on value change (using DDEPoke) 205
B.4. Other Useful Macros 206
C. ODBC Database Concepts **207**
D. Command Line Options **209**
E. Running as a Windows Service (Specified User) **211**
F. Windows Services File for Tunnel/Mirror **213**
G. Error Messages **214**
G.1. Windows Error Numbers 214
G.2. Windows TCP Error Numbers 215
G.3. Windows DDE Error Numbers 216
H. Third-Party Source Licenses **218**
I. Media Source Licenses **220**
J. GNU General Public License **221**
K. GNU Lesser General Public License **227**
I. Skkynet VINE Command Set **235**
acksuccess 238
add 239
alias 240
alive 241
append 242
assembly 243
attribute 244
auth 245
authreload 246
auto_create_domains 247
auto_timestamp 248
bridge 249
bridge_remove 251
bridge_remove_pattern 252
bridge_transform 253
cforce 255
cread 256
create 257
create_domain 258
creport 259
cset 260
cwrite 261
DDEAdvise 263
DDEConnect 264
DDEDisconnect 265
DDEExcelUnicode 266
DDEInit 267
DDEService 268
DDEUnadvise 269

DDEUnadvisePattern 270
DDEUnadvisePoint 271
debug 272
defaultprop 273
delete 274
div 275
domain 276
domains 277
domain_bridge 278
domain_bridge_enable 280
domain_bridge_prefer 281
domain_bridge_refresh 282
domain_bridge_remove 283
dump 284
enable_bridging 285
enable_dde_client 286
enable_dde_server 287
enable_domain_bridging 288
enable_scripting 289
error 290
execute_plugin 291
exit 292
force 293
format 294
get_client_stats 295
heartbeat 296
HistorianAdd 297
HistorianFlags 298
HistorianRemove 299
HistorianSaveConfig 300
HistorianSetConfiguring 301
HistoryGroupAdd 302
HistoryGroupAddPoint 303
HistoryGroupDeadband 304
HistoryGroupDefault 305
HistoryGroupFileTimes 306
HistoryGroupFlushTimes 307
HistoryGroupRemove 308
HistoryGroupStorageTimes 309
ignore 310
ignore_old_data 311
include 312
instance 313
load_config_files 314
load_plugin 315
load_scripts 316
lock 317
log_file 318
log_file_max 319
log_to_file 320
mirror_master 321

mirror_master_2 322
mult 324
private_attribute 325
property 326
quality 327
read 328
report 329
report_domain 330
report_errors 331
request_initial_data 332
save_config 333
secure 334
set 335
set_access 336
set_authoritative 337
set_canonical 338
show_data 339
show_debug_messages 340
show_event_log 341
show_icon 342
show_properties 343
show_script_log 344
subassembly 345
success 346
tcp_service 347
timeout 348
TunnelEnable 349
TunnelEnablePlain 350
TunnelEnableSlave 351
TunnelEnableSSL 352
TunnelPlainPort 353
TunnelSlaveStatus 354
TunnelSSLCert 355
TunnelSSLPort 356
transmit_insignificant 357
type 358
unload_plugin 359
unreport 360
unreport_domain 361
version 362
write 363

II. Obsolete and Unused Commands **365**

asyncsocket 366
authgroup 367
authuser 368
bandwidth_reduce 369
deleted 370
drop_license 371
echo 372
enable_connect_server 373
EnableDDEServer 374

exception_buffer 375
failed_license 376
flush 377
flush_log 378
master_host 379
master_service 380
on_change 381
point 382
qnx_name_attach 383
qnx_receiver 384
readid 385
register_datahub 386
report_all 387
report_datahubs 388
request 389
run 390
script_register 391
script_symbol 392
slave 393
sync 394
taskdied 395
taskstarted 396
using_license 397
warn_of_license_expiry 398
Index ..... **??**
Colophon **406**

Chapter 1. Installation

1.1. System Requirements and Installation

System Requirements

Skkynet VINE is compatible with the following versions of Windows:

- Windows 8 (32-bit & 64-bit)
- Windows 7 (32-bit & 64-bit)
- Windows Server 2012 (64-bit)
- Windows Server 2008 R2 (64-bit)
- Windows Server 2008 (32-bit & 64-bit)
- Windows Vista (32-bit & 64-bit)
- Windows XP SP2 (32-bit & 64-bit)
- Windows Server 2003 SP2 (32-bit & 64-bit)

Installation

To install VINE from an archive downloaded from the Cogent web site, follow these steps:

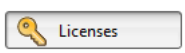
1. Double-click on the program archive `VINE-7.3-xxxxxx-Windows.exe`.
2. Follow the instructions.

Uninstall

1. From the Start menu, select the Control Panel and then choose Add or Remove Programs.
2. Find Skkynet VINE on this list and double-click it.
3. Click the Remove button and follow the instructions.

1.2. Installing Licenses

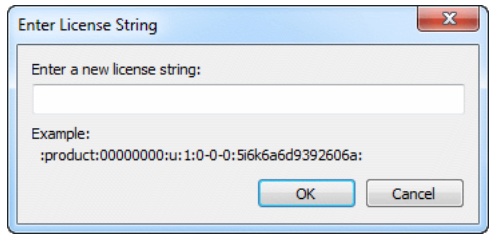
You can install VINE licenses purchased from Cogent using the Licenses option in the Properties window.



The Licenses option lets you view and install licenses for Skkynet VINE. When Skkynet VINE starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, Skkynet VINE switches to license mode, and each connection then requires a license.

Licenses can be entered individually or loaded from a file.

- The Enter a License... button opens the Enter License String window:



Here you can paste or manually enter the text string for the license provided by Cogent. Make sure to include all colon (`:`) characters in the string.

The license string may contain the characters `l` and `1` which can look nearly identical in some type fonts. If possible, it is best to copy and paste the string, rather than retyping it.

- The Load License File... button opens a Windows file selection window. Browse to find the directory and license file that you want to load. License files end with a `.lic` extension. Once you have found the license file, click the Open button to load the file. (Please refer to Configuration and License File Locations in Section 1.5, *Configuration Files* for more information on license file locations.)

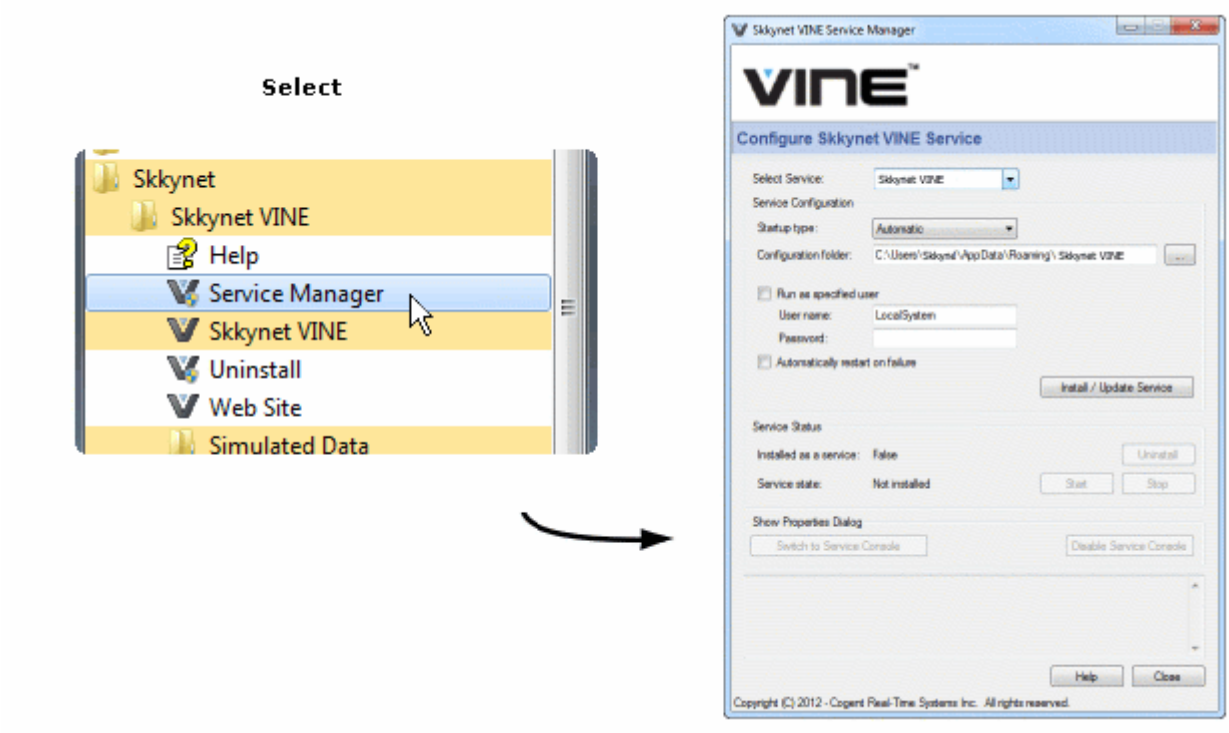
To remove a license from your system, select one or more licenses in the Details window, then click the Remove Selected button.

1.3. Installing VINE as a Service

Skkynet VINE can be installed as a service, using the Service Manager. With this program you can select and configure how VINE runs, change its status, and open the Properties window of Skkynet VINE.

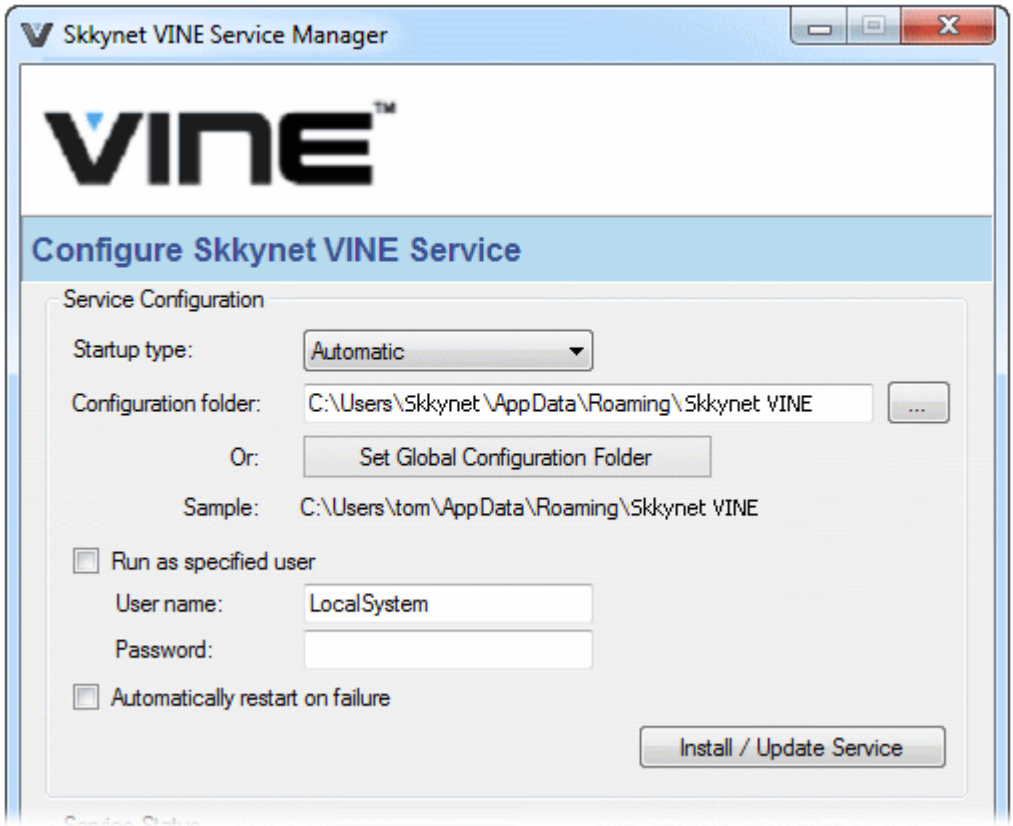
Starting the Service Manager

The Service Manager can be started from the Skkynet VINE program group in the Cogent entry of the Windows Start menu.



Once started, you can choose the service you need to configure from the Select Service dropdown list at the top. Then you can configure, install, and check the status of the service, as well as view the VINE Properties window. The scrolling list at the bottom maintains a record of activities.

Service Configuration



Startup type:

Choose between Automatic, Manual, or Disabled to specify how you want the service to start when Windows starts.

Configuration folder:

Allows you to specify a folder in which to put the configuration files for Skkynet VINE. Typically this does not need to be changed. Please refer to Section 1.5, *Configuration Files* for more information about configuration files.

The Set Global Configuration Folder button lets you specify the configuration file through the registry. This will preserve the configuration directory even after uninstalling as a service.

Run as specified user

It is recommended that you run VINE as the local SYSTEM (`LocalSystem`) user, and leave this box unchecked.

The VINE Properties window is only available when running the service as the local SYSTEM user. If you run as a specified user then you will not be able to access the Properties window to make changes to the VINE while it is running as a service.

However, there are certain situations in which you may need to run it as a specified user. To specify a user other than the local SYSTEM user, enter the User name and Password as applicable, then please see Appendix E, *Running as a Windows Service (Specified User)* for important additional information

Automatically restart on failure

Have the service restart should it fail or be stopped for some reason.

Install/Update Service

When the above configuration is complete, press this button to install VINE as a service. This will also start the VINE service, though on some systems the service may need to be started manually if it doesn't start with the install operation. If the service is running and you make changes to the configuration, pressing this button will cycle through a service shutdown and restart to apply your changes.

Service Status

Installed as a service:

Indicates whether the selected program is installed as a service or not (True or False). The Uninstall button allows you to uninstall VINE as a service.

Service state:

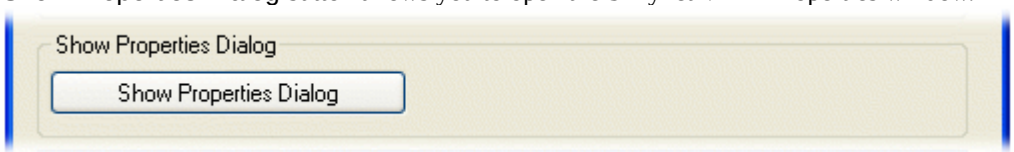
Indicates the run status of the service (Stopped, StartPending, Running, etc.) . The Start and Stop buttons allow you to start or stop the service.

Show Properties Dialog

This option appears differently for different versions of the Windows operating system:

Windows XP and Windows Server 2003

A Show Properties Dialog button allows you to open the Skkynet VINE Properties window.

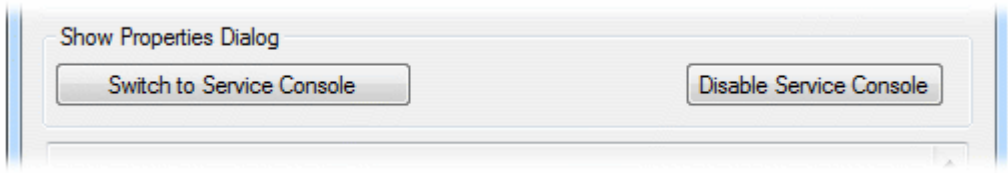


The VINE Properties window is only visible on the primary console of the computer running Skkynet VINE. If you are currently logged in via a remote desktop session, you will see a pop-up dialog indicating that you must be connected to the computer's primary console. You can do this by using the `/admin` or `/console` options on the Microsoft Remote Desktop client.

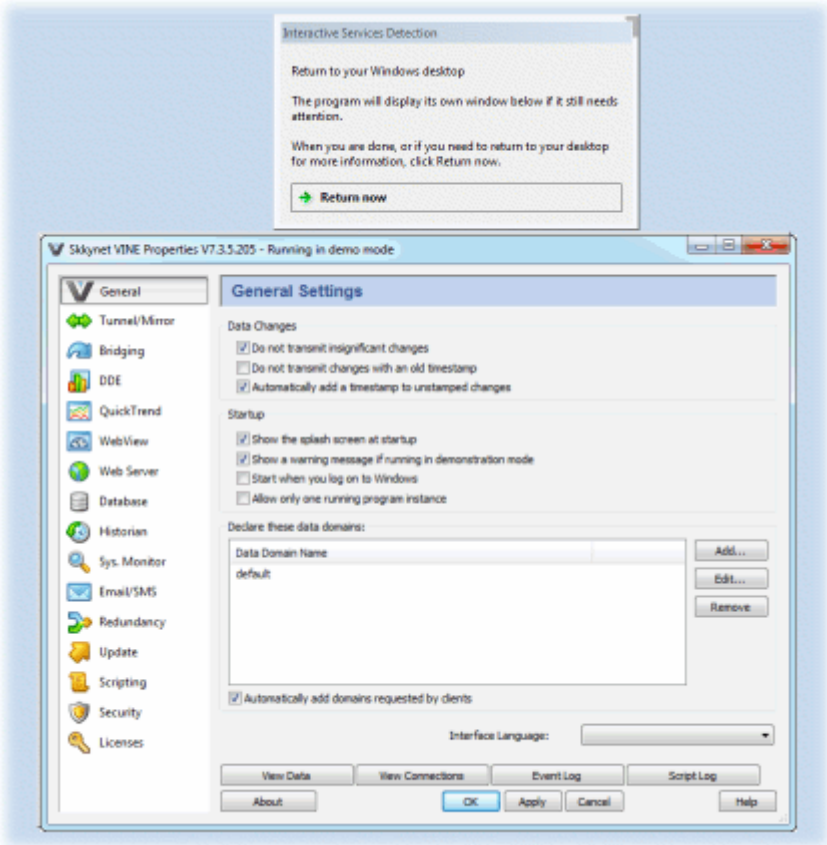
If you are already connected to the primary console, the VINE Properties window and system tray icon will be displayed when you press Show Properties Dialog button.

Windows Vista, Windows 7, Windows 8, Windows Server 2008, and Windows Server 2012

The Switch to Service Console button displays the Windows Service Console, which is where the VINE Properties window appears when running as a service. This allows you to view data and make changes to the DataHub configuration while it is running as a service.



The Service Console is a special display console provided by Microsoft Windows, also known as the "session 0 console". This was introduced in Windows Vista as a security mechanism to limit access to the user interface of high-permission processes. When you switch to the Service Console, your desktop will be hidden and the screen background will change color to indicate the special status of the Service Console.

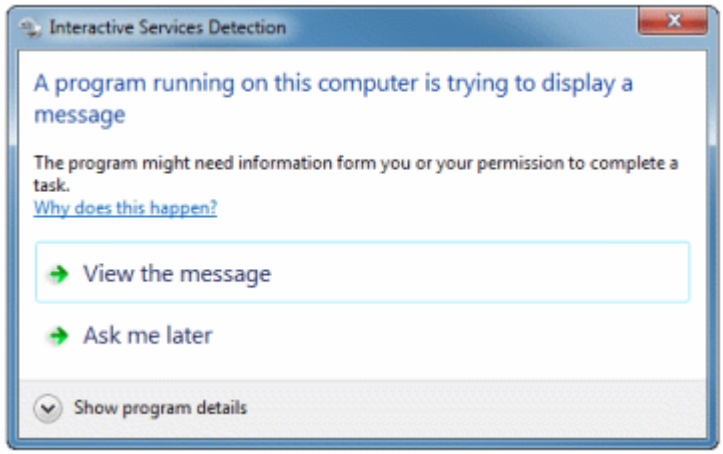


While the Service Console is open, your other applications will continue to run normally, but you will not be able to see or interact with them. A dialog box will be visible while you are viewing the Service Console that will allow you to switch back to your regular desktop at any time. If you have other system services running on your computer that also have user interface windows, those services will also be available to you while viewing the Service Console.

When you have finished viewing or editing the VINE properties, before returning to the normal user console, you should click the Apply and OK buttons to close down the VINE Properties window. Also be sure to close any other VINE windows. This prevents the Windows Interactive Service Detection program from displaying pop-up messages when you return to the normal user console.

Once all the VINE windows are closed, you can return to the normal user console by pressing the Return Now button in the Interactive Service Detection window.

If you forget to close any of the VINE windows while working in the Service Console, Windows will begin popping up messages in the user console telling you there is a program requiring attention in the Service Console.


Interactive Services Detection
A program running on this computer is trying to display a message
The program might need information form you or your permission to complete a task.
Why does this happen?
View the message
Ask me later
Show program details

You can stop these messages and close down the Service Console by clicking on the Disable Service Console button in the VINE Service Manager. Clicking this button will not stop the VINE service.
When you close the Service Manager it will automatically disable the service console. This will stop Windows from periodically displaying the Interactive Services Detection dialog on your system.

1.4. Performing a Silent (Unattended) Install

To perform an unattended install of Skkynet VINE, you can run this command:

```
vine-version number-date-Windows.exe /S /D=c:\program files\cogent
```

`/S`	indicates a silent install
`/D`	indicates the base installation directory

There are certain restrictions with the `/D` argument:

- `/D` must be the last argument on the line
- The path name for `/D` must NOT contain quotes, even if the path contains spaces.
- There must not be spaces around the = sign in the `/D` argument

The installation will create two directories beneath the directory indicated by /D:

- `OPC DataHub` contains the OPC DataHub installation.
- `DataSim` contains the two data simulators.

1.5. Configuration Files

Skkynet VINE comes with several configuration files, including a default configuration file called `default.cfg`. When the program is started for the first time, it uses `default.cfg` to create its primary configuration file, `Skkynet VINE.cfg`. This file gets edited automatically every time you

make changes in the Properties window. Occasionally an experienced user might have reason to edit the `Skkynet VINE.cfg` directly, but in general it should be edited through the Properties window. When you shut down and restart VINE, it reads the configuration in `Skkynet VINE.cfg` to pick up where you left off.

Configuration and License Files Location

Skkynet VINE stores its configuration and license (`licenses.lic`) files in the current user's directory, in a subdirectory named `Skkynet VINE`. Here are the typical locations:

- For Windows XP and Windows Server 2003:

  `C:\Documents and Settings\`*`User Name`*`\Application Data\Skkynet VINE\`

- For Windows Vista, Windows 7, Windows 8, Windows Server 2008, and Windows Server 2012:

  `C:\Users\`*`UserName`*`\AppData\Roaming\Skkynet VINE\`

If you ever need to back up the VINE configuration, you need to copy everything in the `Skkynet VINE` directory (above). To copy the configuration from one VINE to another PC, you need to copy everything except your license file.

The `Application Data` or `AppData` directory might be a hidden directory.

If there is no private configuration when VINE starts, it will search for configuration files and license files from previous versions in the application installation directory, and copy them to the user's private configuration. If there are no old configuration files, VINE will copy all the current configuration files from the application installation directory. Thereafter, changes to Skkynet VINE's properties (made through the Properties window) will only change the user's private configuration.

A user can modify this behaviour in two ways with the following command line options:

1. Provide the `-H` *`home`* option which will indicate to VINE that it should store the private configuration and license files in the directory specified by *`home`*, rather than in a subdirectory of `Application Data`. Skkynet VINE will still search for previous and default configuration files and licenses in the application installation directory as described above.
2. Provide the `-U` option which will indicate to VINE that it should not create private configuration files for each user, but store its configuration in the application installation directory.

If both `-H` and `-U` are specified, the `-U` flag is ignored.

Custom Configuration Files

When Skkynet VINE starts up, you may wish to have certain points and data structures get created immediately. To do this, you can create one or more custom configuration files. These files must be listed in the bottom of the Scripting option of the Properties Window.

Creating and editing custom configuration files should only be attempted by experienced users. If you do create a custom configuration file, we strongly recommend that it only be used for creating points and data structures, and not for standard configuration commands, such as those that are created and modified through entries in the Properties window. Doing otherwise could result in irregular behavior in Skkynet VINE.

The following sets of commands are the only ones that should be used in a custom configuration file:

General commands allowed in config files

create
set
cset
write
cwrite
force
cforce
add
mult
div
lock
quality
secure
append
dump
include

Model-related commands allowed in config files

alias
assembly
attribute
defaultprop
instance
private_attribute
property
subassembly
type

Only allowed in Windows config files

execute_plugin
load_plugin
unload_plugin
load_scripts
show_data
show_debug_messages
show_event_log
show_icon
show_properties
show_script_log

Custom configuration files can be created using a text editor like Notepad, and are written in Lisp syntax. They should be put in the same directory as the Skkynet VINE executable, such as `C:\Program Files\Cogent\Skkynet VINE\` (it may be different for your installation). Each entry of the file contains either a command or a comment. Comments are marked with a semicolon character (`;`) at the beginning of each line.

For more information about commands and their syntax, please refer to Reference I, *Skkynet VINE Command Set*.

A small custom configuration file might look like this:

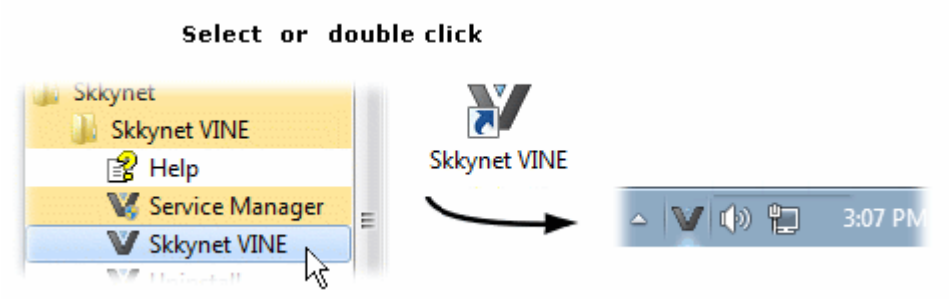
```
;;; Create some points in the default data domain
(create default:Point1)
(create default:Point2)
(create default:Point3)

;;; Assign values and confidences to the points
(set default:Point1 5 100)
(set default:Point2 67.234 100)
(set default:Point3 Hello 100)


;;; Create a new data domain
(create_domain NewDomain)

;;; Create some points in that data domain
(create NewDomain:Point1)
```

```
(create NewDomain:Point2)
(create NewDomain:Point3)

;;; Assign a value and confidence to the point
(set NewDomain:Point1 "A string" 100)
(set NewDomain:Point2 95 100)
(set NewDomain:Point3 3.1519 100)
```

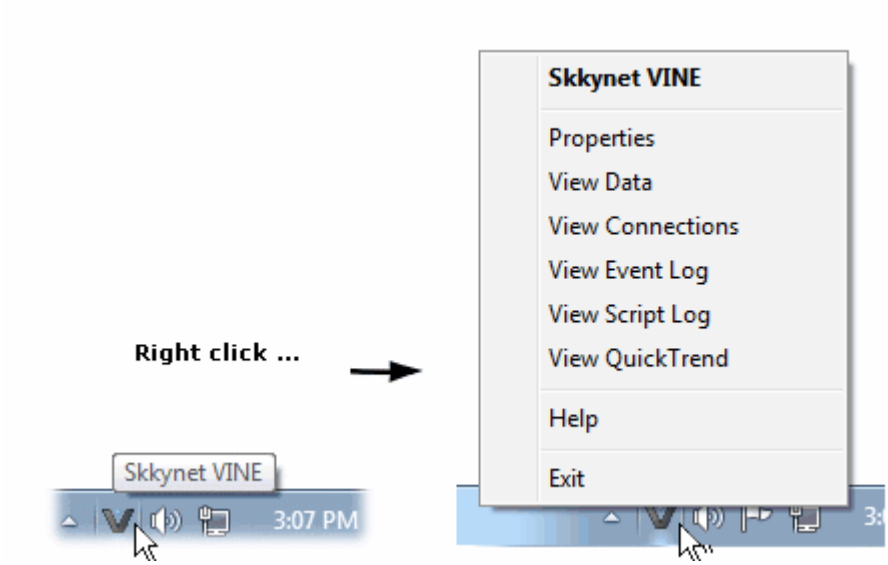
Chapter 2. Getting Started

2.1. Running Skkynet VINE

To run VINE, select the program using the Windows Start menu, or double click the desktop icon.

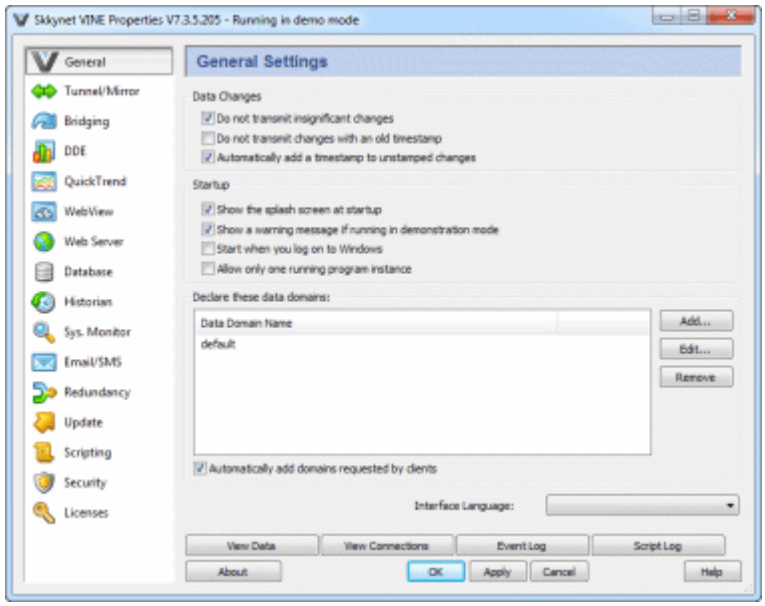


Once VINE is started, it runs in the background and puts a VINE icon in the system tray.

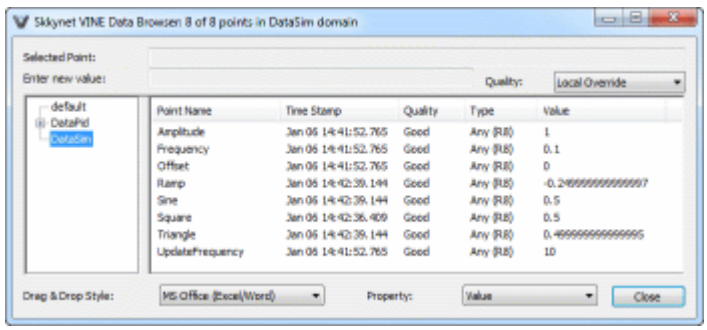


If you right-click on that icon, you will get a small pop-up menu with several options that let you open the following windows:

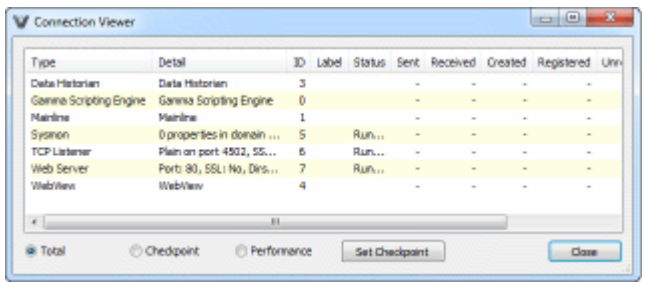
Touch-screen users: press the icon for about one second. When you release, the pop-up menu will appear.



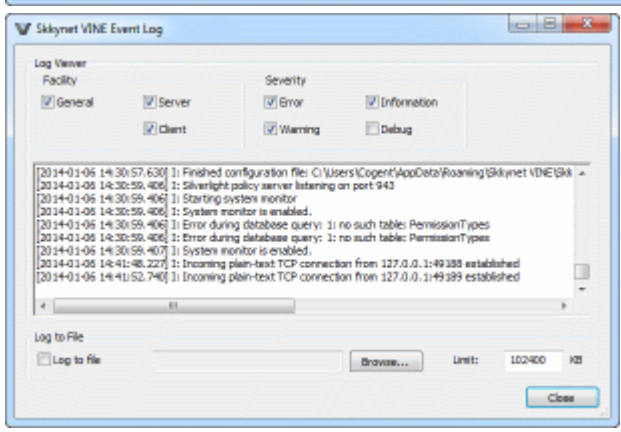
Properties window



Data Browser

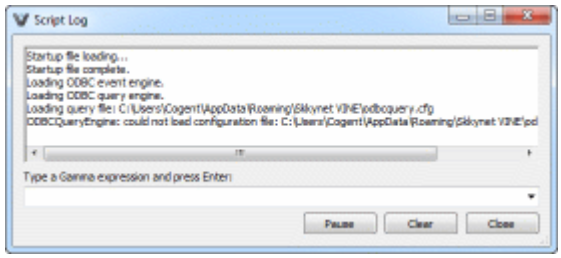


Connection Viewer

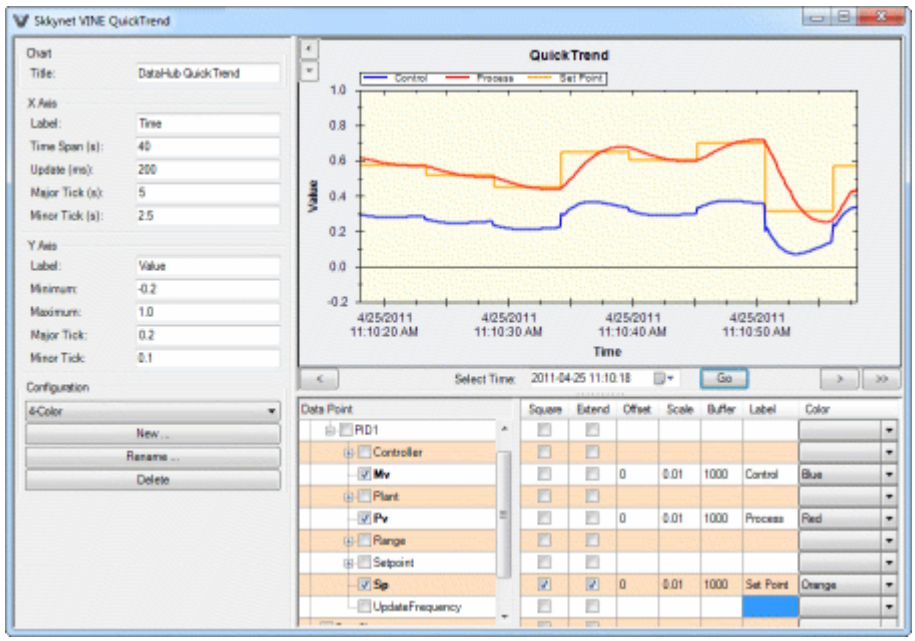


Event Log

Script Log



QuickTrend

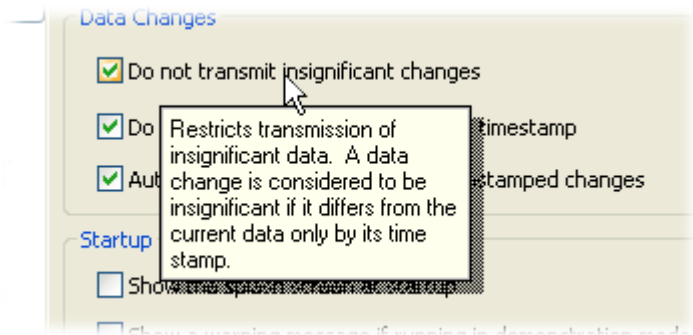


The pop-up menu also lets you exit VINE.

It is possible to run VINE with command-line options. Please refer to Appendix D, *Command Line Options* for more information.

Pop-up Help

You can get pop-up help in many parts of the Properties window by right-clicking the mouse over buttons or text.


Data Changes
Do not transmit insignificant changes
Restricts transmission of insignificant data. A data change is considered to be insignificant if it differs from the current data only by its time stamp.
Startup

Exit

You can terminate VINE by right-clicking Skkynet VINE icon in the system tray, and selecting Exit

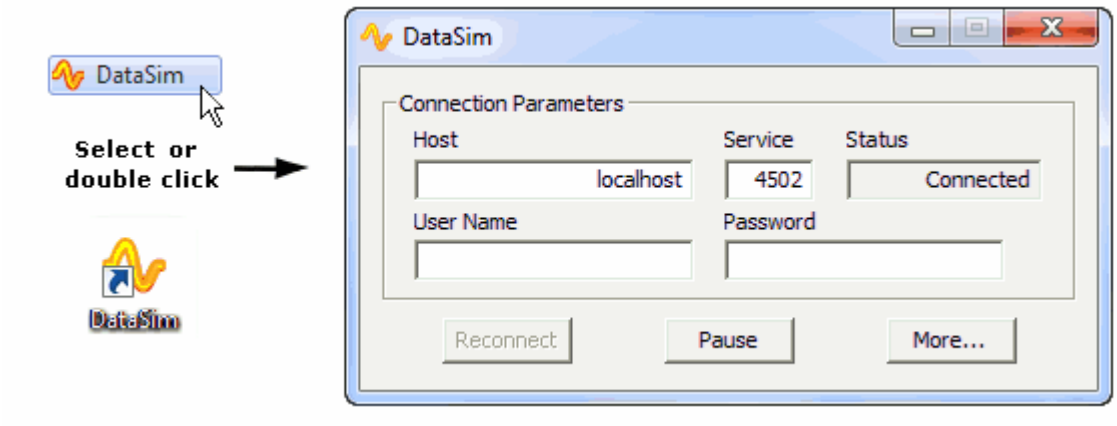
from the pop-up menu. After a few seconds the icon should disappear, indicating that VINE has terminated.

You must explicitly exit VINE to terminate it. Otherwise it continues to run in the background even if you close the Properties, Data Browser, and Event Log windows.

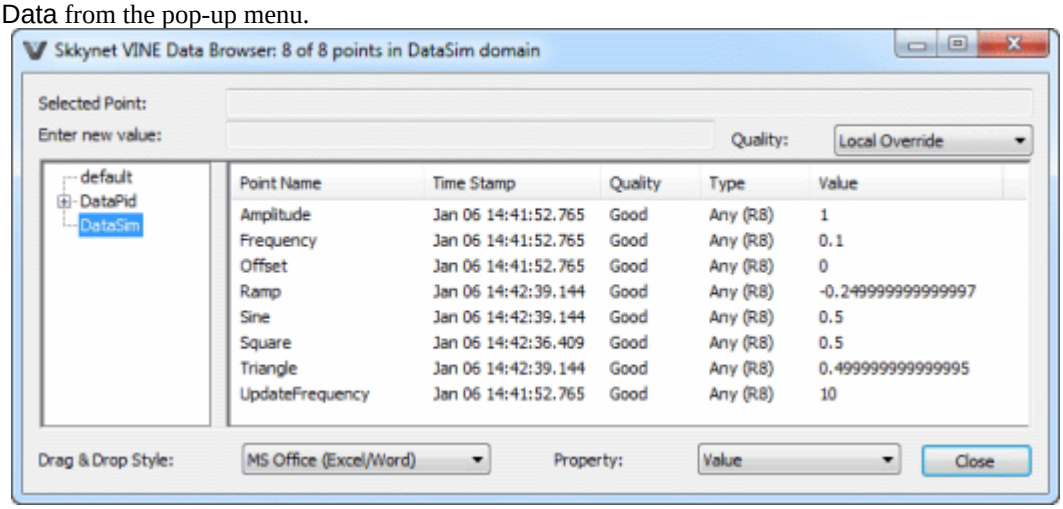
2.2. Test with simulated data

There is a data-generating program that comes with Skkynet VINE called DataSim. You can run DataSim locally to create data for various connection scenarios.

1. Start Skkynet VINE if it isn't already running.
2. Start DataSim using the Windows Start menu, or by double clicking the desktop icon.



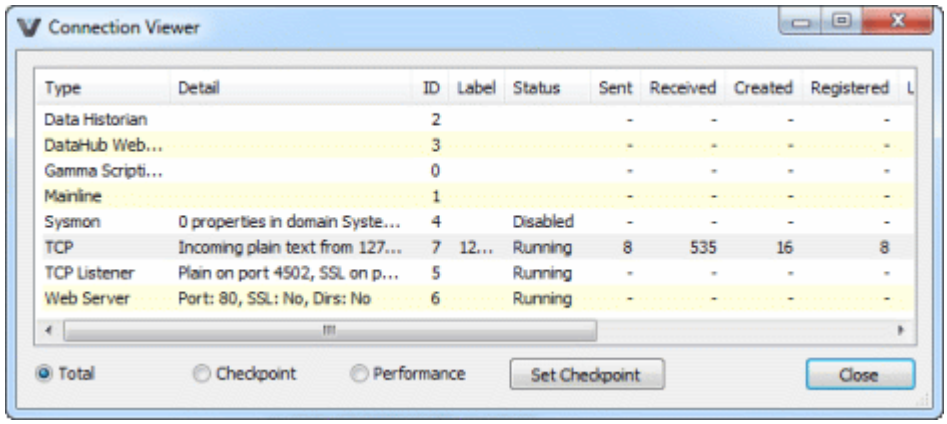
3. Open the VINE Data Browser by right clicking on the VINE system-tray icon and choosing View Data from the pop-up menu.



4. Select the DataSim data domain in the left-hand pane of the window.

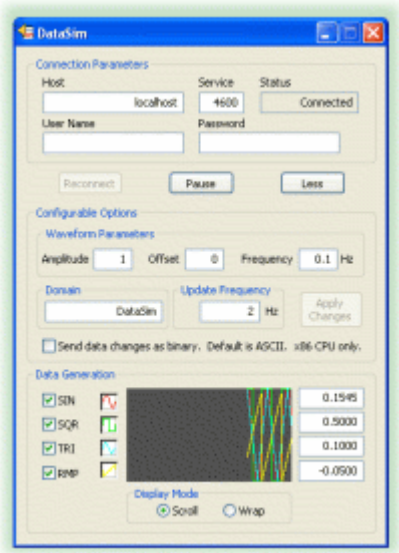
   The Data Browser window should fill with simulated data updating in real time. The four updating data points are `Ramp`, `Sine`, `Square`, and `Triangle`.

5. You can also click the View Connections button near the bottom of the Properties window to see your currently configured connections:



The Connection Viewer shows all active connections in Skkynet VINE.

6. You can click the More... button in DataSim to access some options for changing the data feed.



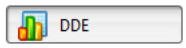
Briefly, you can change the Configurable Options and click the Apply Changes button to apply them. The Waveform Parameters and Update Frequency are all points in Skkynet VINE, and the corresponding points change their values in the Data Browser as you make the changes. Changing the Data Domain from `DataSim` will yield no results until custom data domains are configured for VINE.

If you shut down DataSim, its points will still appear in VINE until it is shut down and restarted. Please refer to Section 13.1, *Data Points* for more information on creating and deleting points.

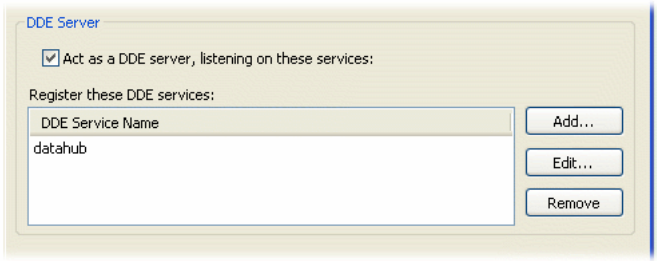
Now you are ready to start using VINE.

2.3. Test with Excel

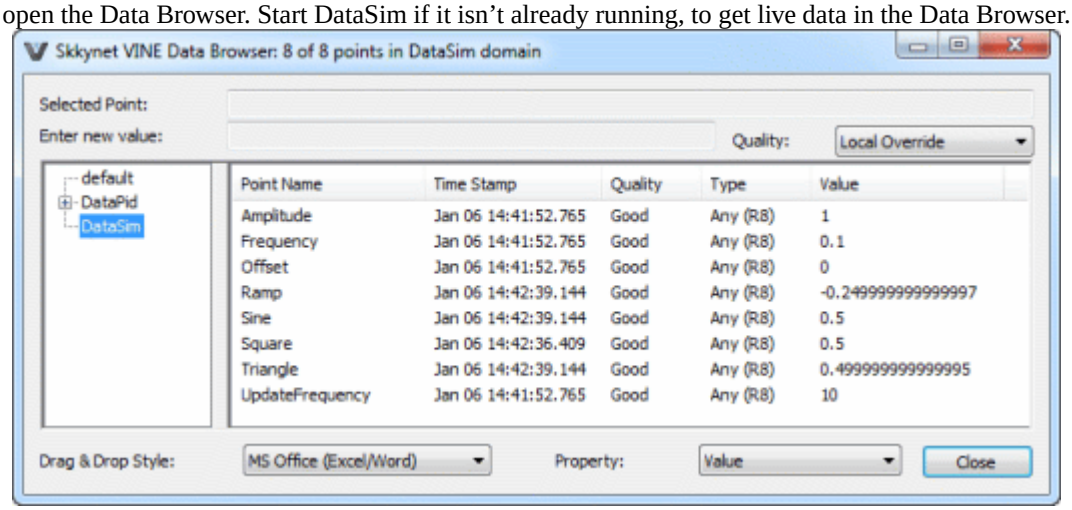
To test Skkynet VINE by putting live data into Excel, follow these steps:

1. Right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select DDE 

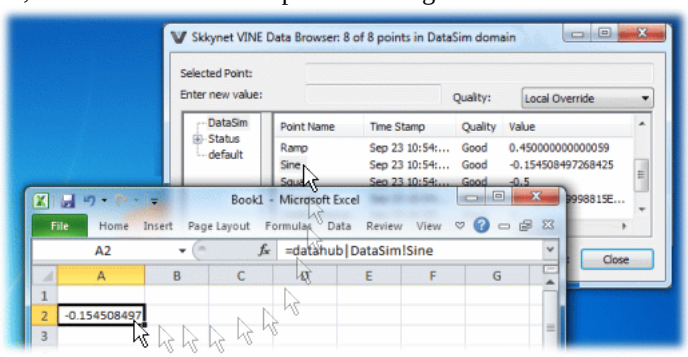
.



3. Ensure that the box Act as a DDE server is checked, and that the name `vine` appears in the DDE Service Name area. If not, click the Add... button and add the name `vine`.
4. Click OK to close the Properties window.
5. Right click on the Skkynet VINE system-tray icon and choose View Data from the pop-up menu to open the Data Browser. Start DataSim if it isn't already running, to get live data in the Data Browser.


Skkynet VINE Data Browser: 8 of 8 points in DataSim domain
Selected Point:
Enter new value:
Quality:
Local Override
default
DataPid
DataSim
Point Name
Time Stamp
Quality
Type
Value
Amplitude Jan 06 14:41:52.765 Good Any (R8) 1
Frequency Jan 06 14:41:52.765 Good Any (R8) 0.1
Offset Jan 06 14:41:52.765 Good Any (R8) 0
Ramp Jan 06 14:42:39.144 Good Any (R8) -0.249999999999997
Sine Jan 06 14:42:39.144 Good Any (R8) 0.5
Square Jan 06 14:42:36.409 Good Any (R8) 0.5
Triangle Jan 06 14:42:39.144 Good Any (R8) 0.499999999999995
UpdateFrequency Jan 06 14:41:52.765 Good Any (R8) 10
Drag & Drop Style:
MS Office (Excel/Word)
Property:
Value
Close

6. Ensure that the Drag & Drop Style at the bottom of the Data Browser is set to MS-Office (Excel/Word).
7. Open an Excel worksheet.
8. In the Data Browser, click on the label for a point and drag it into the Excel worksheet.



You should see the data update in the worksheet at the same rate it is updating in Skkynet VINE.

You can select multiple points for drag and drop by using **Shift**-click or **Ctrl**-click.

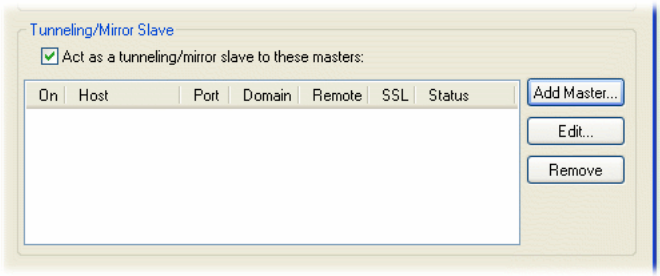
You can drag and drop timestamps and other attributes of a point using the Property dropdown list. Please refer to Drag and Drop Style and Property in the Data Browser section for more details.

2.4. Connect to remote data

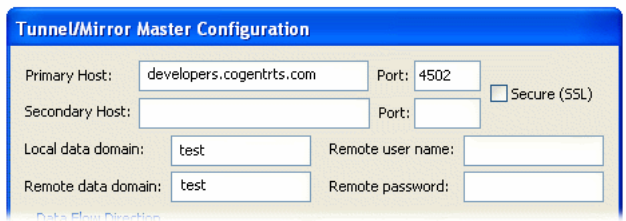
Cogent has a VINE running that makes live test data available over the Internet. To set up Skkynet VINE to receive that test data, just follow these steps:

1. Right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.
3. In the Tunnelling/Mirror Slave section, check the Act as a tunnelling/mirroring slave to these masters box.

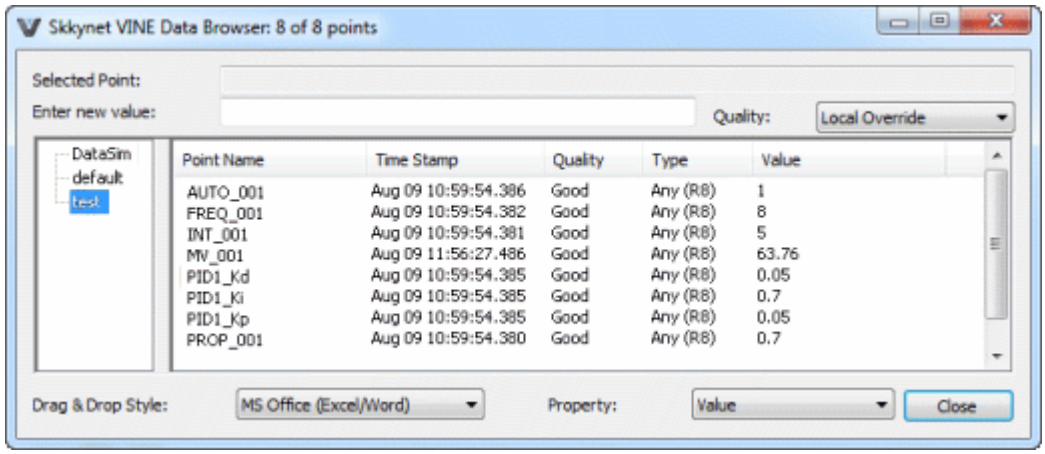


4. Click the Add Master... button to open the Tunnel/Mirror Master Configuration window:



5. Fill in these entry fields at the top as follows:
   - **Primary Host:** `developers.cogentrts.com`
   - **Local data domain:** `test`
   - **Remote data domain:** `test`

   There is no need to make or change any other entries.
6. Click OK to close the Tunnel/Mirror Master window, and then click Apply in the Properties Window.
7. Open the Data Browser and click the test data domain name in the left-hand pane of the window.



The Data Browser window should show some data updating in real time. Any delays in updates to these points are due to slow network speed or high traffic volumes.

You can also use the Connection Viewer to see all active connections in Skkynet VINE.

For more information on tunnelling/mirroring, please refer to Section 15.2, *Tunnel/Mirror*.

Chapter 3. Excel Connections

You can use Skkynet VINE to put data into Excel, and to write data from Excel back to VINE.

The following sections explain how to drag and drop live data into Excel, how to configure Skkynet VINE to receive data, and how to use Excel macros for sending and receiving data between Excel and VINE.

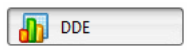
3.1. Getting Data into Excel

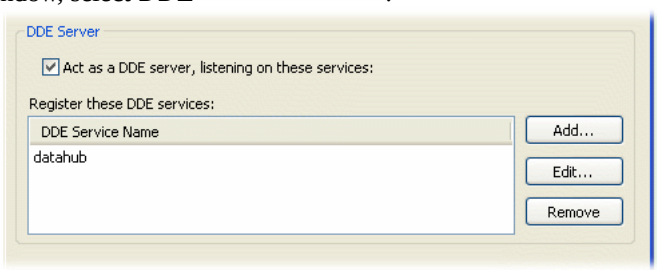
Before starting, to see any results you will have to ensure that you have some kind of data being fed into Skkynet VINE. If your system isn't set up for this yet, you can create a local data feed by following the steps outlined in Section 2.2, *Test with simulated data*.

There are two ways to get data into Excel from Skkynet VINE: by setting up a **DDEAdvise** loop to receive data automatically, or by using a **DDERequest** command from a macro to read data. Deciding which to use depends on your situation. We suggest you become familiar with both. For more information about DDE and these commands, please refer to Section 13.3.1, *DDE Protocol* and Appendix A, *DDE Overview*.

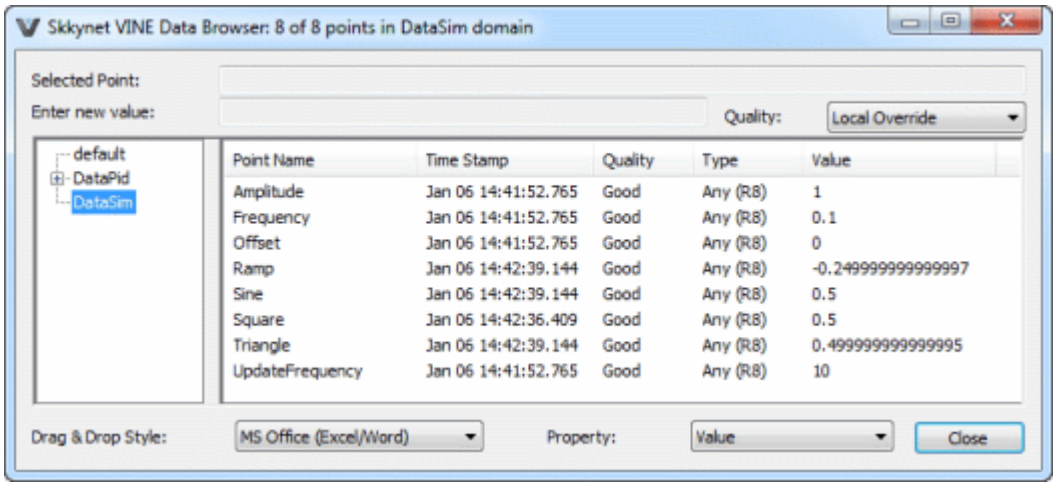
3.1.1. Method 1 - Drag and Drop using DDEAdvise

The easiest way to get data into Excel is to drag and drop point names from the VINE Data Browser directly into the Excel spreadsheet. This automatically sets up a **DDEAdvise** loop between Excel and VINE. **DDEAdvise** loops update automatically so you will always see the latest data in your spreadsheet.

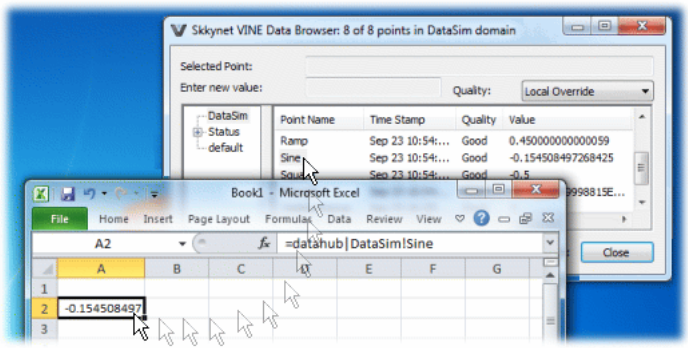
1. Right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select DDE 

.


3. Ensure that the box Act as a DDE server is checked, and that the name `vine` appears in the DDE Service Name area. If not, click the Add... button and add the name `vine`.
4. Click OK to close the Properties window.
5. Right click on the Skkynet VINE system-tray icon and choose View Data from the pop-up menu to open the Data Browser.


Skkynet VINE Data Browser: 8 of 8 points in DataSim domain
Selected Point:
Enter new value:
Quality:
Local Override
default
DataPid
DataSim
Point Name
Time Stamp
Quality
Type
Value
Amplitude Jan 06 14:41:52.765 Good Any (R8) 1
Frequency Jan 06 14:41:52.765 Good Any (R8) 0.1
Offset Jan 06 14:41:52.765 Good Any (R8) 0
Ramp Jan 06 14:42:39.144 Good Any (R8) -0.249999999999997
Sine Jan 06 14:42:39.144 Good Any (R8) 0.5
Square Jan 06 14:42:36.409 Good Any (R8) 0.5
Triangle Jan 06 14:42:39.144 Good Any (R8) 0.499999999999995
UpdateFrequency Jan 06 14:41:52.765 Good Any (R8) 10
Drag & Drop Style:
MS Office (Excel/Word)
Property:
Value
Close

6. Ensure that the Drag & Drop Style at the bottom of the Data Browser is set to MS-Office (Excel/Word).
7. Open an Excel worksheet.
8. In the Data Browser, click on the label for a point and drag it into the Excel worksheet.



You should see the data update in the worksheet at the same rate it is updating in Skkynet VINE.

You can select multiple points for drag and drop by using **Shift**-click or **Ctrl**-click.

You can drag and drop timestamps and other attributes of a point using the Property dropdown list. Please refer to Drag and Drop Style and Property in the Data Browser section for more details.

If your data displays but does not update, you might need to change your settings in Excel. Please refer to Chapter 18, *Troubleshooting* for more information.

When you save and close a spreadsheet connected to Skkynet VINE, and then attempt to reopen it, you may get one or more messages, depending on your security settings in Excel, or other circumstances. Here's a summary of each message, and what to do:

This document contains macros. Enable them?

Click Enable Macros.

This workbook contains links. Update them?

Click Update. If VINE is already running, all the links should then update automatically. If VINE is not running, you will get a #REF! entry in each cell that has an advise loop established with VINE, and the next message (see below) will probably appear.

Remote data not accessible. Start VINE?

Click No. At this point the best thing to do is close the worksheet, start VINE manually, and then reopen the worksheet. When you update the spreadsheet (see above) this time you won't get any #REF! entries. If, instead of No you click Yes at this point, VINE will not start, but instead generate an error message, and Excel may even crash later on.

3.1.2. Method 2 - Excel Macros using DDERequest

Sometimes, you may prefer to manually read data into your spreadsheet, rather than use a **DDEAdvise** loop to constantly accept new values. It may be that you intend to print reports only a couple of times a day and don't need to see every point change in between. You can have Excel read specific data points from Skkynet VINE at your request by triggering the **DDERequest** command from within a macro.

Using **DDERequest** within a macro gives you complete control over when Excel reads new point values, and lets you read several data points at one time. To run the macro, it is convenient to link it to a control button. This is explained in Add a Control Button.

3.1.2.1. Create a macro

1. Open a spreadsheet.
2. From the Tools menu, select Macro, and then Macros....
3. In the Macro Name: field of the Macro dialog box, type the name **`GetInput`**, and press the Create button.
4. In the Visual Basic text entry window that comes up, edit the macro to read as follows:

```
'
' GetInput Macro
'
Sub GetInput()
    mychannel = DDEInitiate("vine", "default")
    Application.Worksheets("Sheet1").Activate
    newval = DDERequest(mychannel, "my_pointname")
    Sheet1.Cells(2, 3) = newval
    DDETerminate mychannel
End Sub
```

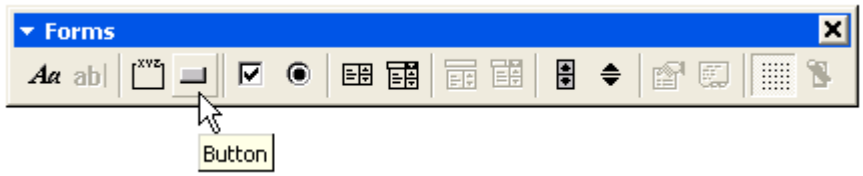
Use the name of your data point from Skkynet VINE for *`my_pointname`*.

We use cell `C2` in this example. If you need to use another cell, you will have to replace `(2, 3)` with the row and column numbers of the cell you wish to use.

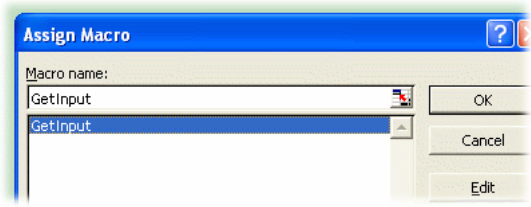
5. Save and close the Visual Basic text entry window.

3.1.2.2. Add a Control Button

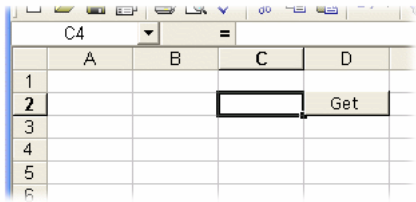
1. Activate the Forms toolbar by clicking on the View menu and selecting Toolbars, and then Forms.



2. Click on the button icon, and then click in cell D2. (We use this cell in our example, but you can choose another cell if you'd like.) An Assign Macro window should appear.
3. Select GetInput and click OK.



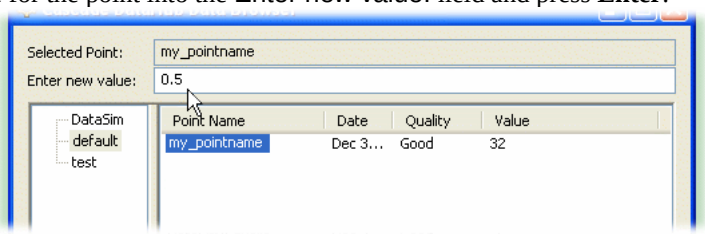
4. Change the label on the button to "Get".



5. For appearance, you can move the button, resize it with the handles, and change the size of the text by right-clicking on it and selecting Format Control.
6. Save the spreadsheet.

3.1.2.3. Receive the data

1. Now you're ready to receive the data. Open the VINE Data Browser if it is not already open, go to the `default` data domain, and find the name of the point.
2. Click on the point to highlight it. The point name should appear in the Selected Point: field at the top of the Data Browser.
3. Type a new value for the point into the Enter new value: field and press **Enter**.



4. Go to Excel and click the Get button. You should see the data update each time you click the button.

3.2. Getting Data out of Excel

There are two ways to get data out of Excel and into Skkynet VINE:

1. **Configure a DDEAdvise loop** in VINE that instructs Excel to send data automatically to the DataHub any time a value changes.

   The data is sent immediately to VINE, every time the specified cell or range changes. This does not allow any kind of sanity check or safeguard on the data being sent, but in some cases it may be desirable to have Excel emit data automatically.

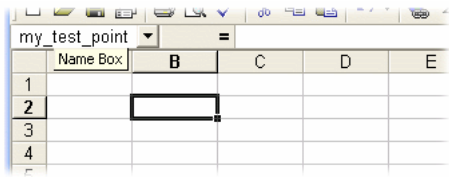
   Each time data is sent for one point (data item), it is sent for all points. This can tie up your network if you have a large number of points. If you need to send data for a large number of cells, you can reduce this effect and reduce CPU load by sending a range that contains the cells.

2. **Write a macro in Excel** that uses the **DDEPoke** command to 'push' data from Excel to VINE. This allows you to define exactly when the data is sent to VINE.

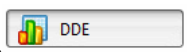
3.2.1. Method 1 - Configuring DDEAdvise loops in Skkynet VINE

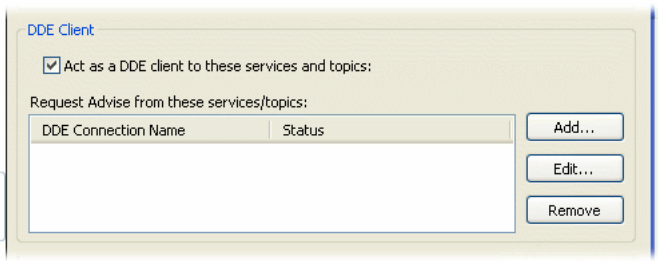
The quickest and easiest method to get data from Excel to VINE is to configure one or more **DDEAdvise** loops in VINE to automatically receive data from Excel, which is acting as a DDE server.

1. Open an Excel spreadsheet.
2. Choose a cell or range to hold the data you want to put into VINE. You will need to refer to your cell or range by row and column number, or by a name. For example, the cell B2 can be referred to as R2C2, or by giving it a name.



   To name a cell or range, select it and enter a unique name the box just above the first column of the worksheet. Then save the worksheet.

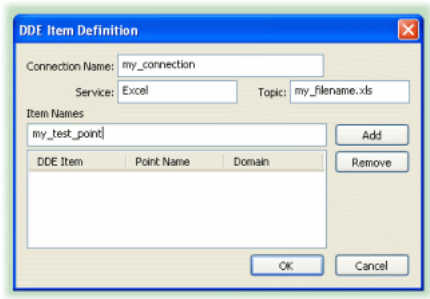
3. Start Skkynet VINE if it isn't already started, and open the Properties Window (by right-clicking on the VINE icon in the Windows system tray and selecting Properties).
4. Click the DDE button. 



5. Make sure the Act as DDE client box is checked.

   For best performance, ensure that a DDE server (in this case, Excel) is running when using VINE as a DDE client. A DDE client can consume substantial system resources trying to connect if a DDE server is not available.

6. Click the Add button. This opens the DDE Item Definition window where you can add Excel as a new DDE service.



7. Type in the following information:
   - **Connection Name:** choose a name to identify this connection. It must be unique among all DDE connections.
   - **Service:** type in **`Excel`**.
   - **Topic:** type the name of your worksheet file. In Windows XP, this name is the same as what is shown after the dash in the title-bar of the Excel spreadsheet. More recent versions of Windows might not show the complete name in the title bar. In any case, you must use the complete file name, so if the worksheet is named, "`Book1`", then your Topic is simply `Book1`, but if the worksheet is named `Test.xls` then your Topic needs to be `Test.xls`.

     If you want to link to a cell or range which is not on the first sheet in the workbook, you need to put the filename in square brackets, followed by the sheet name. For example, if your worksheet name is `Test.xls`:

Sheet in workbook	Service	Topic to enter
The first sheet	**`Excel`**	**`Test.xls`**
An unnamed sheet (e.g. Sheet2)	**`Excel`**	**`[Test.xls]Sheet2`**
A named sheet (e.g. StockData)	**`Excel`**	**`[Test.xls]StockData`**

   - **Item Names:** type in the row and column numbers or the name you entered as the cell or range name in Excel (in step 2 above).
8. Click the Add button. The fields DDE Item, Point Name and Data Domain are then added to the list of items associated with this **DDEAdvise** loop. You can continue to add points for other cells in your spreadsheet or click OK to close the dialog.

   The DDE Item is associated with a point in VINE. You can change the Point Name and Data Domain to anything you want by double clicking on the name and typing a new name. When you click OK, the new point will be created in the DataHub.
9. Click OK to close the DDE Item Definition window. The new **DDEAdvise** loop is added to the list.
10. Click the Apply button for your changes to take effect. Once you have done this, you should see the **DDEAdvise** loop connection Status change to Connected.
11. Open the Data Browser by right clicking the VINE icon in the system tray and selecting View Data
12. With the default data domain chosen, scroll down to see the name of the point.
13. In Excel, type a number into the cell or range you named in step 2, and press **Enter**. You should see the data update in the Data Browser.

Although this is an easy way to send data from Excel, it is not the most efficient when you have a large number of points to transmit. Whenever Excel transmits a data point using **DDEAdvise**, it also transmits the current value of every other point associated with any **DDEAdvise** loop. Where you have a large number of cells to update, we have found it to be much more efficient to transmit the data as Excel ranges. In your **DDEAdvise** loop, define a range of cells that contains the data you want to transmit. Using Excel ranges will reduce the load on the computer and make it easier to configure your application.

Another option for reducing the load on the computer when transmitting a large number of points is to write an Excel macro that uses **DDEPoke** to transmit data on a timed basis, say once a second. Information on how to write a macro in Excel to do this is given below.

When you save and close a spreadsheet connected to Skkynet VINE, and then attempt to reopen it, you may get one or more messages, depending on your security settings in Excel, or other circumstances. Here's a summary of each message, and what to do:

This document contains macros. Enable them?

Click Enable Macros.

This workbook contains links. Update them?

Click Update. If VINE is already running, all the links should then update automatically. If VINE is not running, you will get a #REF! entry in each cell that has an advise loop established with VINE, and the next message (see below) will probably appear.

Remote data not accessible. Start VINE?

Click No. At this point the best thing to do is close the worksheet, start VINE manually, and then reopen the worksheet. When you update the spreadsheet (see above) this time you won't get any #REF! entries. If, instead of No you click Yes at this point, VINE will not start, but instead generate an error message, and Excel may even crash later on.

3.2.2. Method 2 - Writing Excel macros that use the DDEPoke command

Writing an Excel macro is perhaps the most flexible and efficient way to send data from Excel to Skkynet VINE. By using the **DDEPoke** command in an Excel macro you have complete control over exactly when the data is transmitted. We will also explain how you can write an Excel macro to transmit multiple points at the same time (see Additional Pointers for more details).

In our example, we have chosen to 'add a control button' to run the macro, but you could also run your macro on a timed interval to produce an automatic update on a cycle that you control.

3.2.2.1. Create a macro

1. Open a spreadsheet.
2. From the Tools menu, select Macro, and then Macros....
3. In the Macro Name: field of the Macro dialog box, type the name **`SendOutput`**, and press the Create button.
4. In the Visual Basic text entry window that comes up, edit the macro to read as follows:

```
'
' SendOutput Macro
'
Sub SendOutput()
    mychannel = DDEInitiate("vine", "default")
    Application.Worksheets("Sheet1").Activate
    Call DDEPoke(mychannel, "my_pointname", Cells(4, 3))
    DDETerminate mychannel
```

```
End Sub
```

Use the name of your data point from Skkynet VINE for *`my_pointname`*.

We use cell `C4` in this example. If you need to use another cell, you will have to replace `(4, 3)` with the row and column numbers of the cell you wish to use. You can also name a range to send multiple values as an array.

5. Save and close the Visual Basic text entry window.

3.2.2.2. Add a Control Button

This explanation is illustrated in Section 3.1.2.2, *Add a Control Button*. We repeat the text briefly here.

1. Activate the Forms toolbar by clicking on the View menu and selecting Toolbars, and then Forms.
2. Click on the button icon, and then click in cell D4. (You can choose another cell if you'd like.) An Assign Macro window should appear.
3. Select SendOutput and click OK.
4. Change the label on the button to "Send".
5. Save the spreadsheet.

3.2.2.3. Send the data

1. Now you're ready to send the data. Open the Skkynet VINE Data Browser if it is not already open, go to the default data domain, and find the name of the point.
2. In Excel, type a number in cell C4 (or the cell or range you assigned the macro to) and press **Enter**.
3. Click the Send button.
4. You should see the data update.

3.2.2.4. Additional Pointers

- To reduce CPU for large amounts of data, send arrays of data using ranges instead of sending the data for each cell as a separate point.
- If you are using Unicode characters in strings for **DDEPoke** commands, you should check the Accept non-English characters in Excel strings (slower) button in the DDE option of the Properties window.

  This will cause Excel to send your strings of Unicode characters correctly, although slower than numerical data.

- The **DDEInitiate** and **DDETerminate** commands that are used to open and close DDE links between applications are also very CPU expensive. When sending variables at frequent intervals it is more efficient to open a DDE channel at the beginning of the session and close it when you are finished. Here are two suggestions:

1. Send multiple points within a single set of **DDEInitiate** and **DDETerminate** commands. For example:

```
'
' Cascade Multiple Writeback macro
'
Sub Cascade_Writeback_Many()
      mychannel = DDEInitiate("vine", "default")
      Application.Worksheets("variables").Activate
      DDEPoke(mychannel, "pointname1", Cells(1,2))
      DDEPoke(mychannel, "pointname2", Cells(2,2))
      DDEPoke(mychannel, "pointname3", Cells(3,2))
      DDEPoke(mychannel, "pointname4", Cells(4,2))
      DDEPoke(mychannel, "pointname5", Cells(5,2))
      DDEPoke(mychannel, "pointname6", Cells(6,2))
      DDETerminate mychannel
End Sub
```

   In this example the worksheet named `variables` contains six variables (`pointname1` through `pointname6`) that we wish to send to Skkynet VINE. The **DDEInitiate** command opens the channel, then all six variables are sent to VINE before the link is closed.

2. Create a separate 'open' and 'close' macro for the worksheet, and place the **DDEInitiate** and **DDETerminate** commands in those macros. This will keep communication to VINE open for the whole time the worksheet is open. The only drawback is that your data transmission could get interrupted (see below).

- If you need to send data continually from Excel to Skkynet VINE you may run into problems using **DDEInitiate** and **DDEPoke**. When you open a DDE channel using the **DDEInitiate** statement, and follow it with several **DDEPoke** statements, there is a chance that the DDE channel may fail after some time. For this reason, if you need to keep a DDE channel open for an extended period of time, we suggest that you attempt to deal with DDE errors within the macro.

3.3. Working with Ranges

Skkynet VINE can send and receive the data contained in an entire range of an Excel spreadsheets. This data is treated as an *array*, a two-dimensional range of cells as rows and columns. The array can be as big as necessary (within point size limits), or as small as a single cell—at least one row and one column.

Data Format

Excel transmits array data as a tab-and-newline delimited text string of values. Each value in a row is separated by a tab, and each row is separated by a newline character. The string does not contain any information concerning the source range of the array within the spreadsheet.

3.3.1. Getting a Range out of Excel

There are two methods of transmitting a range, or array data, from Excel to Skkynet VINE. These exactly match the mechanisms used for individual point data: DDEPoke and DDEAdvise.

Using DDEPoke with a Macro

A DDEPoke command can be issued by Excel to send data to Skkynet VINE based on a trigger within Excel. For this to work, Skkynet VINE needs to be configured to act as a DDE server and have registered at least one service name. An Excel macro can then issue a DDEPoke to that service, along with a

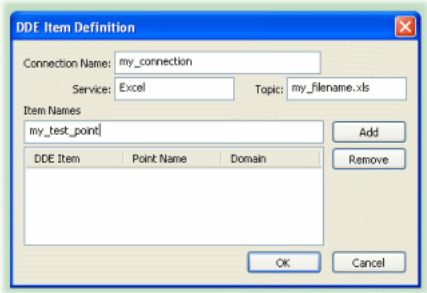
Skkynet VINE data domain name (the DDE topic), a point name (the DDE item) and a value. If the value is of type `Range` then Excel will automatically format the value as a tab-and-newline separated string.

Example: See the definition of the `PutData` function in the Excel macro coding examples below.

Using a DDE Advise Loop

When sending data from Excel to Skkynet VINE using a DDE advise loop, Excel acts as the DDE server and VINE acts as the client. To create the advise loop:

1. Open the Skkynet VINE Properties Window (by right-clicking on the VINE icon in the Windows system tray and selecting Properties).
2. Click the DDE button.
3. Make sure the Act as DDE client box is checked.
4. Click the Add button. This opens the DDE Item Definition window.



5. Type in the following information:
   - **Connection Name:** choose a name to identify this connection. It must be unique among all DDE connections.
   - **Service:** type in **`Excel`** (case is not important).
   - **Topic:** type the name of your worksheet file, including the `.xls` extension, like this: *`my_filename`*`.xls`.
   - **Item Names:** These create a mapping between Excel cells and ranges, and Skkynet VINE point names. You may specify a single cell in *`r1c1`* format, a range of cells in *`r1c1`*`:`*`r2c2`* format, a cell name, or a range name as the DDE Item name. For example:

`r2c5`	- accesses the cell `E2` (second row, fifth column)
`r3c3:r5c9`	- accesses the range `C3:I5`
`MyRange`	- accesses the cell or range that is named `MyRange`

6. Click the Add button. The fields DDE Item, Point Name and Data Domain should automatically fill in with some values.

   Check the names in the Point Name and Data Domain columns. If either of them is not what you need, double-click it to select it, and change it.
7. Click OK to close the DDE Item Definition window. The fields DDE Connection Name and Status in the Properties Window should now be filled in as well.
8. Click OK to close the Properties Window.
9. Enter some values in the range of the spreadsheet you have defined. You should see the array in the Data Browser change accordingly.

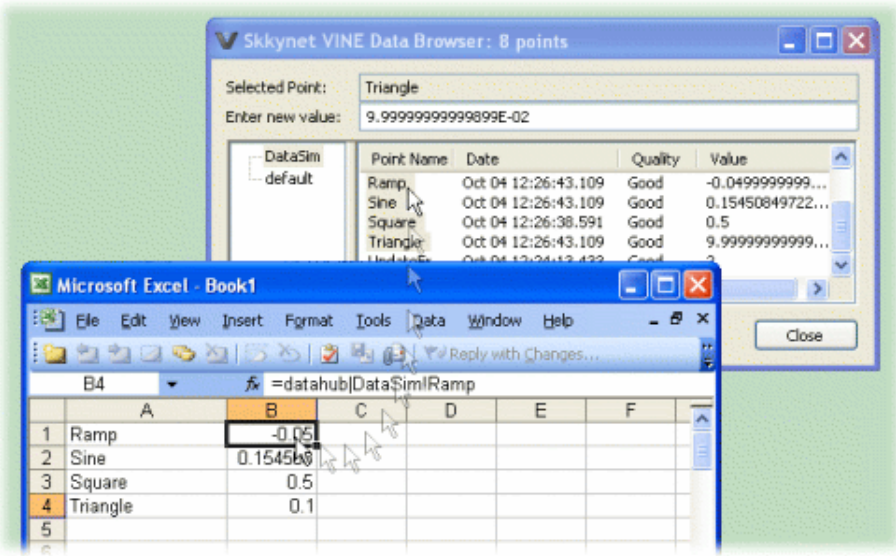
3.3.2. Getting a Range into Excel

There are two ways to drag and drop data into Excel to create a range, using DDE advise loops. Or you can use DDE Request and macros.

Drag and drop a group of points into Excel

Here is how you can collect a group of points in VINE and drag them all into Excel, where the data for each point occupies a unique cell.

1. With Skkynet VINE and DataSim running, open the Data Browser.
2. Select a group of points in the Data Browser.
3. Drag the point names into Excel.



You should see the data updating in the cells.

You can drag and drop point names, timestamps, and other attributes of a point using the Property dropdown list. Please refer to Drag and Drop Style and Property in the Data Browser section for more details.

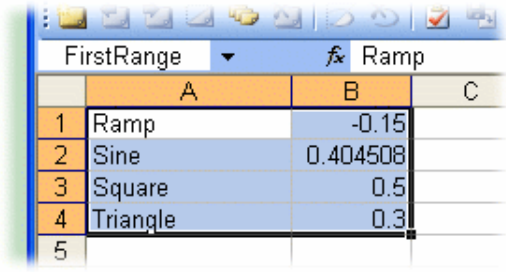
Drag and drop an array into Excel

Here is how you can take a single point in VINE whose value is an array, and have each value in the array occupy a unique cell in Excel.

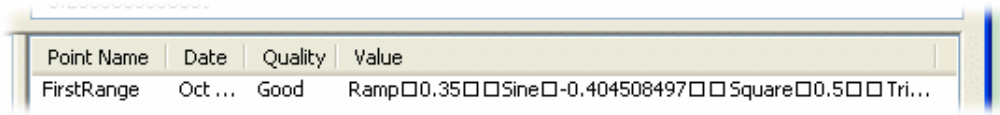
To demonstrate this, we are going to first combine the two procedures shown above to create an array in VINE

Make an array

1. Select a range in Excel, such as created in Drag and drop a group of points into Excel above, and in the name box at the top left corner, enter the name **`FirstRange`**.



2. In the Skkynet VINE Properties Window, select the DDE option and make sure the Act as DDE client box is checked. Then click the Add button.
3. In the DDE Item Definition window type in the following information:
   - **Connection Name:** type in **`Ranges`**.
   - **Service:** type in **`Excel`**.
   - **Topic:** type in **`Book1`**, or the name of your worksheet file including the `.xls` extension.
   - **Item Names:** Type in **`FirstRange`**.
4. Click the Add button. The fields DDE Item and Point Name should be `FirstRange`, and the Data Domain should be `default`.
5. Click OK to close the DDE Item Definition window, and in the Properties Window click OK to close it as well.
6. Open the Data Browser and go to the `default` data domain. You should see the point `FirstRange`, with a value like this:

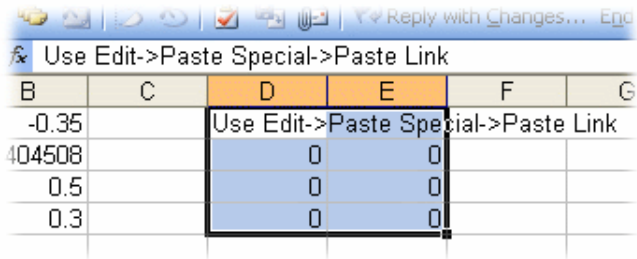


The single boxes separate values in a row, and the double boxes separate rows. The array is now ready to put into Excel.

Drag and drop the array

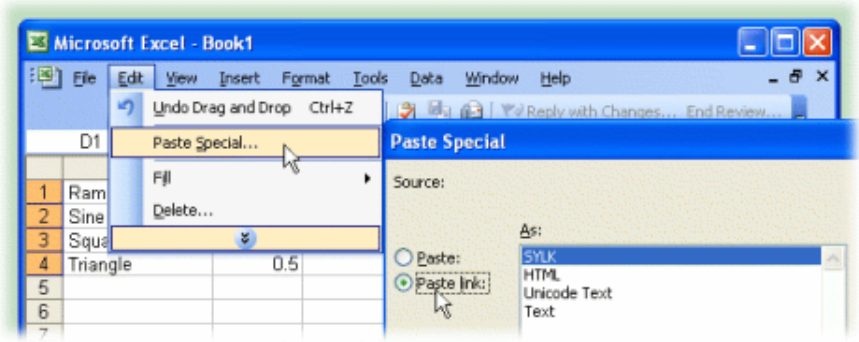
For simplicity's sake we are going to just put the same array back into Excel.

1. Click on the `FirstRange` point name, and drag it into Excel, dropping it in cell D1.



The values don't start updating right away because you have to tell Excel how to paste in the link.

2. Go to the Edit menu and select Paste Special.



3. Select Paste link and click OK. The cells should fill with the correct, updating data.

Using DDE Request in Excel

If you are creating macros in Excel to read data from Skkynet VINE, you can use the DDERequest function call. This will return an array type value that can be written directly into any range in the spreadsheet. If the array data is larger in any dimension than the range into which it is written, then extra data in the array is discarded. If the array data is smaller than the target range then extra cells in the range are filled by repeating the data in the array. See below for an Excel macro that dynamically determines the target range to ensure that all array data is entered into the spreadsheet with no duplication.

3.3.3. Sample Excel Macros for Arrays

The following macros represent the entire macro set for a simple test spreadsheet that reads and writes a single array point in Skkynet VINE. The two functions `GetData` and `PutData` can be attached to buttons on a spreadsheet for easy testing. The `PutData` subroutine contains two alternative representations of the source range, one of which is commented out in the macro.

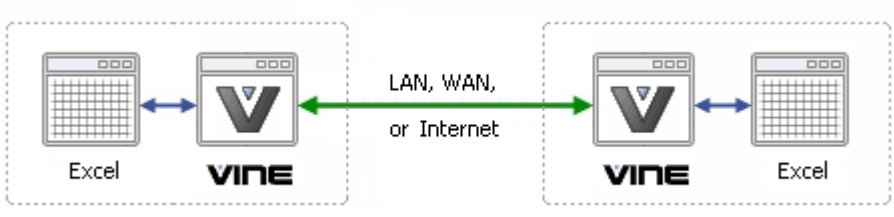
```
Sub GetDataArray(Channel As Integer, SheetName As String, DataPoint As String, _
                 StartRow As Integer, StartCol As Integer)
    Dim NRows As Integer, NCols As Integer

    ' This sub performs a DDERequest for DataPoint in the DDE Channel and reads in a tab
    ' delimited array with carriage returns at the end of each line. It then fills a range
    ' of cells with the data. The native format for Excel data is tab delimited text with a
    ' carriage return at the end of each row of data. If we assign this type of data to a
    ' range of cells using the FormulaArray function, Excel automatically parses the data
    ' and fills it into the specified range.  The real trick here is to ensure that the
    ' range is the same size as the incoming data, so we do not have to know the size
    ' a priori.

    DataArray = DDERequest(chan, DataPoint)   ' request DataPoint from Channel

    ' find the upper row and column bounds for the variant array

    If StartCol = 0 Then StartCol = 1   ' Starting column where data will go in our sheet
    If StartRow = 0 Then StartRow = 1   ' set the starting row
    NCols = 1                           ' set default number or columns to 1
    On Error Resume Next                ' ignore errors (error occurs if array has
                                        '                one dimension)

    ' get upper bound of the array columns
    ' the following line will generate an error if the array is only a one dimensional array
    ' We just skip this, and use the default 1
    NCols = UBound(DataArray, 2)

    On Error GoTo 0                     ' allow errors
    NRows = UBound(DataArray, 1)        ' get upper bound of array y dimension

    NRows = NRows + StartRow - 1        ' add offset from StartRow - this is the ending row
```

```
    NCols = NCols + StartCol - 1        ' add offset from StartCol - this is the ending col

    ' the following line fills up the cells in the range starting in "StartCol:StartRow" to
    ' "Nrows:Ncols" with the data from the variant array
    Sheets(SheetName).Range(Cells(StartRow, StartCol), Cells(NRows, NCols)) = DataArray
End Sub

Sub PutDataArray(Channel As Integer, SheetName As String, DataPoint As String, _
                 StartRow As Integer, StartCol As Integer, NRows As Integer, _
                 NCols As Integer)
    DDEPoke Channel, DataPoint, Sheets(SheetName).Range(Cells(StartRow, StartCol), _
            Cells(StartRow + NRows - 1, StartCol + NCols - 1))
End Sub
Sub PutDataRange(Channel As Integer, DataPoint As String, DataRange As Range)
    DDEPoke Channel, DataPoint, DataRange
End Sub

Sub GetData()
'
'   This is a test function assigned to a button.  It reads a test point into
'   an arbitrarily sized matrix starting at A10
'
    Dim chan As Integer
    chan = DDEInitiate("vine", "default")
    GetDataArray chan, "Sheet1", "TestArray", 10, 1
    DDETerminate (chan)
End Sub

Sub PutData()
'
'   This is a test function assigned to a button.  It writes a 3 row x 5 column
'   area of Sheet1 into a single data point in VINE.  You can use either
'   PutDataArray or PutDataRange, depending on how you wish to specify the range.
'
    Dim chan As Integer
    chan = DDEInitiate("vine", "default")
    'PutDataArray chan, "Sheet1", "TestArray", 1, 1, 3, 5
    PutDataRange chan, "TestArray", Sheets("Sheet1").Range("A1:E3")
    DDETerminate (chan)
End Sub
```

Chapter 4. Tunnel/Mirror for Networking

4.1. Introduction

When talking about data networks, *tunnelling* means to encapsulate one protocol inside another, to allow it to be sent more easily and/or securely across the network. *Mirroring* is the ability to mirror part of all of the data set between the two ends of the tunnel, in real time. Skkynet VINE supports both, simultaneously, so to make it simple we refer to simply "tunnelling" in this document for establishing a tunnel/mirror connection across a network. For more information about these terms, please refer to Section 15.2, *Tunnel/Mirror*.



Using Skkynet VINE for tunnelling means:

- Connect equally easily across a LAN, WAN, or the Internet.
- Real-time data transfer.
- Complete and secure data access.
- Simple set up.

Setting up Skkynet VINE for tunnelling is a simple, 2-step configuration that can be done in a few minutes. All you need to do is:

1. Configure VINE to connect to a data server or client on each machine.
2. Configure the tunnelling master or slave on each machine.

4.2. Configuring the tunnelling master

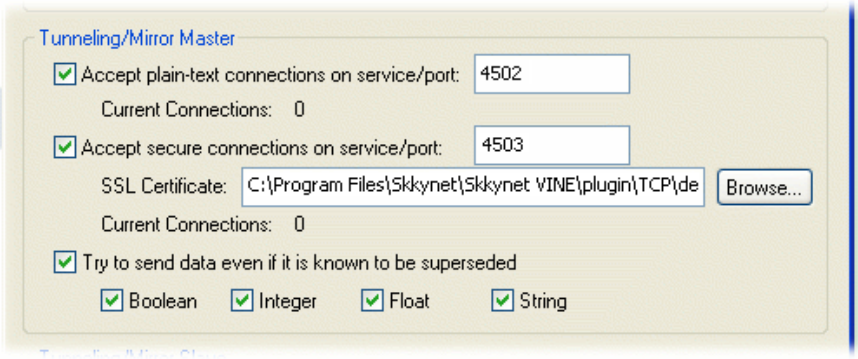
We identify the two tunnelling/mirroring VINEs as *master* and *slave*. The only difference between the master VINE and slave VINE is that the slave initiates the connection. Once the connection is established, they function exactly the same. It is possible for an instance of VINE to be both tunnelling/mirroring slave and master simultaneously—acting as a slave to one or more VINEs and a master to one or more others.

A typical scenario for multiple tunnels is a hub-and-spoke arrangement, in which a number of remote locations share data with a central location. In such a scenario, the central VINE is configured as tunnelling master, and each of the remote VINEs as a tunnelling slave.

Configure VINE as tunnelling master

1. Right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.



3. In the Tunnelling Master section, you can configure plain-text or secure tunnelling. Ensure that at least one of these is checked. If you want to change any of the other defaults, please refer to Section 15.2, *Tunnel/Mirror* for more information.

   To optimize throughput, un-check the Try to send data even if it is known to be superseded option. This will allow VINE to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

4. Click OK to close the Properties window.

The master side of the tunnelling connection is now ready, and you can now configure the slave machine(s).

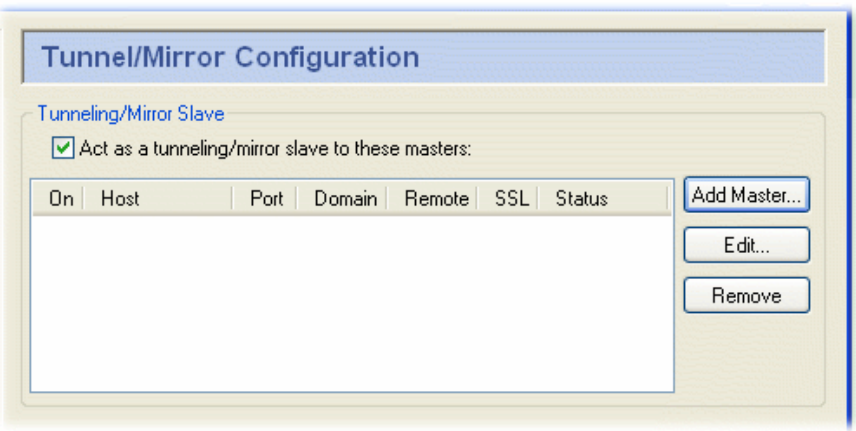
4.3. Configuring a tunnelling slave

We identify the two tunnelling/mirroring VINEs as *master* and *slave*. The only difference between the master VINE and slave VINE is that the slave initiates the connection. Once the connection is established, they function exactly the same. It is possible for an instance of VINE to be both tunnelling/mirroring slave and master simultaneously—acting as a slave to one or more VINEs and a master to one or more others.

Configure Skkynet VINE as tunnelling slave

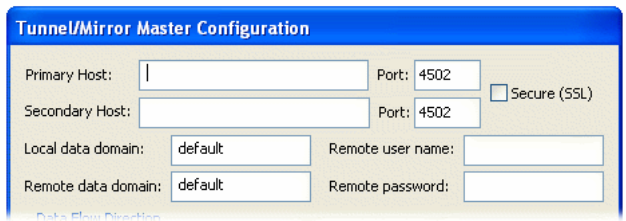
1. Right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.



3. Check the box Act as a tunnelling/mirror slave to these masters.

4. Click the Add Master... button to assign a master to this slave. The Tunnel/Mirror Master Configuration window will open:



5. Type in the following information:

   - **Primary Host:** the name or IP address of the computer running the tunnelling master VINE.
   - **Port:** the port number or service name for this host. You should use default port number (`4502`) unless you have changed the entry in the master VINE.
   - **Secondary Host:** gives you the option to have an alternate host and service/port number. On startup or after a network break, VINE will search first for the primary host, then for the secondary host, alternating between primary and secondary until a connection is made. If no secondary host is specified, the connection will be attempted on the primary host only.

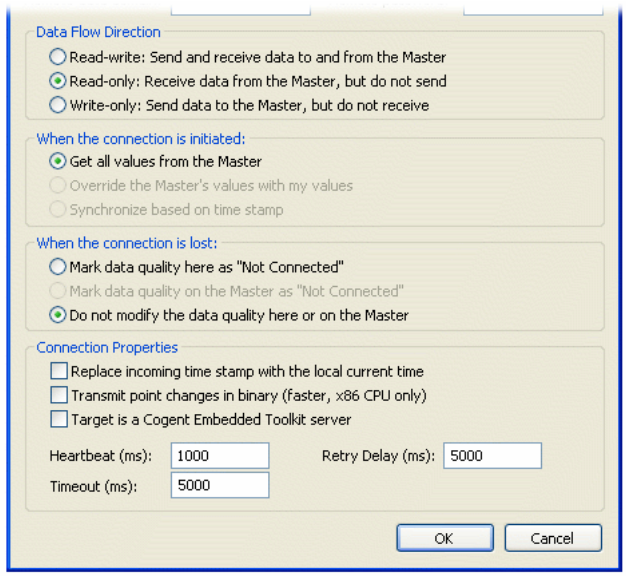
     This feature is not recommended for redundancy because it only checks for a TCP disconnect. The VINE Redundancy feature, on the other hand, provides full-time TCP connections to both data sources, for instantaneous switchover when one source fails for any reason. There is no need to start up the OPC server and wait for it to configure its data set. You can also specify a preferred source, and automatically switch back to that data source whenever it becomes available. By contrast, the primary and secondary host in the tunnel can act as a primitive form of redundancy, but will only switch on a connection failure at the TCP level, which is only one sort of failure that a real redundancy pair must consider.

   - **Local data domain:** The data domain in which you plan to receive data.
   - **Remote data domain:** the master VINE data domain from which you plan to receive data. Point names will be mapped from the remote data domain (on the master VINE) into the local data domain (on this VINE), and vice versa.

     Unless you have a good reason for making these different, we recommend using the same data domain name on both VINEs for the sake of simplicity.

   There is a VINE running on Cogent's server that you can connect to for testing. Here are the parameters you will need to enter for it:
   - **Primary Host:** `developers.cogentrts.com`
   - Port: `4502`
   - **Local data domain:** `test`
   - **Remote data domain:** `test`

6. You now have several options for the mirrored connection.


Data Flow Direction
Read-write: Send and receive data to and from the Master
Read-only: Receive data from the Master, but do not send
Write-only: Send data to the Master, but do not receive
When the connection is initiated:
Get all values from the Master
Override the Master's values with my values
Synchronize based on time stamp
When the connection is lost:
Mark data quality here as "Not Connected"
Mark data quality on the Master as "Not Connected"
Do not modify the data quality here or on the Master
Connection Properties
Replace incoming time stamp with the local current time
Transmit point changes in binary (faster, x86 CPU only)
Target is a Cogent Embedded Toolkit server
Heartbeat (ms): 1000
Retry Delay (ms): 5000
Timeout (ms): 5000
OK
Cancel

a. **Data Flow Direction:** lets you determine which way the data flows. The default is bi-directional data flow between slave and master, but you can effectively set up a read-only or write-only connection by choosing that respective option.

To optimize throughput, check the Read-only: Receive data from the Master, but do not send option. Only do this if you actually want a read-only connection. If you do not require read-write access, a read-only tunnel will be faster.

b. **When the connection is initiated:** determines how the values from the points are assigned when the slave first connects to the master. There three possibilities: the slave gets all values from the master, the slave sends all its values to the master, or the master and slave synchronize their data sets, point by point, according to the most recent value of each point (the default).

c. **When the connection is lost:** determines where to display the data quality as "Not Connected"—on the master, on the slave, or neither.

If you have configured When the connection is initiated as Synchronize based on time stamp (see above), then this option must be set to Do not modify the data quality here or on the Master to get correct data synchronization.

d. **Connection Properties** gives you these options:

- **Replace incoming timestamp...** lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.
- **Transmit point changes in binary** gives users of x86 CPUs a way to speed up the data transfer rate. Selecting this option can improve maximum throughput by up to 50%.

For more information, please refer to Section 14.1, *Binary Mode Tunnel/Mirror (TCP) Connections*.

- **Target is a Cogent Embedded Toolkit server** allows this slave to connect to an Embedded Toolkit server rather than to another VINE.
- **Heartbeat** sends a heartbeat message to the master every number of milliseconds specified here, to verify that the connection is up.

- **Timeout** specifies the timeout period for the heartbeat. If the slave VINE doesn't receive a response from the master within this timeout, it drops the connection. You must set the timeout time at least twice the heartbeat time.

  To optimize this setting for slow networks, please refer to Section 14.2, *Tunnel/Mirror (TCP) Connections for Slow Networks*.

- **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

7. Click OK to close the Tunnel/Mirror Master window. The fields in the Tunnelling Slave table of the Properties Window should now be filled in.
8. Click the Apply button in the Properties Window. If the master VINE is running, this VINE should establish the tunnelling connection, and the Status should display `Connected`. You can view the data with the Data Browser, or view the connection with the Connection Viewer.

With data from the tunnelling master now available, you can now feed it to any VINE client, such as an Excel worksheet, as described in Section 3.1, *Getting Data into Excel*.

Chapter 5. Bridging

5.1. Introduction

Bridging means connecting two data servers together so they can access each other's data, even when neither of them can act as a client.

In addition to bridging two data servers on a single computer, Skkynet VINE offers advanced bridging capabilities that let you:

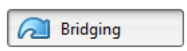
- Bridge data servers over a network connection, by tunnelling.
- Bridge between any number of data servers.
- Scale, convert, or normalize the data as it is bridged from one server to the other, with built-in linear transformations.
- Define even more complex relationships between points in code using VINE Scripting.

You can configure the bridges you need using the Bridging option in the Properties Window.

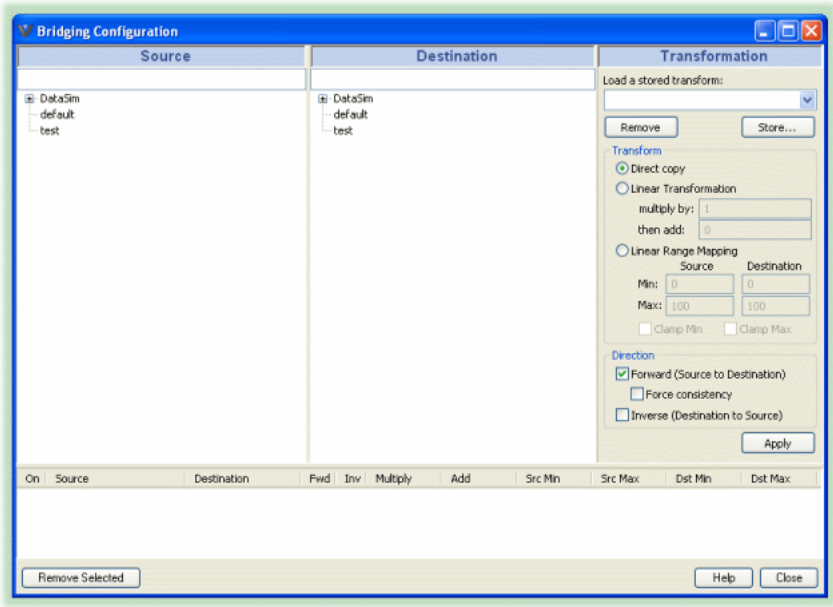
5.2. Configuring Bridges

It is easy to configure Skkynet VINE to bridge existing points—just point and click. If necessary, you can quickly configure linear transformations and specify the direction of data flow of the bridge. And should you want to create a new point for a bridge, it's just few more clicks of the mouse. All configuration and any changes are done on the fly, taking effect as soon as you click the Apply button.

5.2.1. Point-to-point configuration

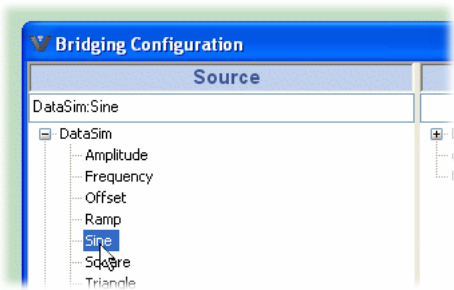
1. With VINE running, right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select Bridging 

.
3. Click the Configure Bridges button. The Bridging Configuration window will open.



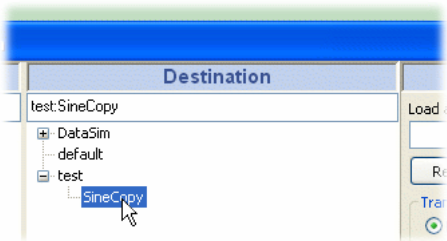
The three top panes in this window correspond to the three basic steps in making the configuration: specify a source, a destination, and any desired transformations. The horizontal pane across the bottom shows the bridges that exist on the system.

4. From the tree diagram in the Source panel, select a source point that you want to bridge.



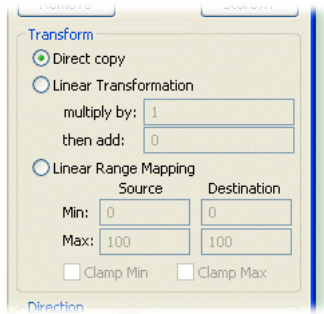
For example, if you have the DataSim program running, you can select the point Sine in the DataSim data domain. The name of the point gets automatically entered in the field at the top of the panel. Alternatively, you can type the name of the point in the entry field.

5. In the tree diagram in the Destination panel, select a destination point.



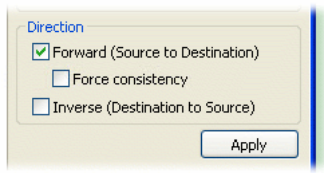
When you select a destination point, its name gets automatically entered in the field at the top of the panel. Or you can type the name of a point in the entry field.

6. Specify direct copy or transformation.



To make a direct copy, just leave the default Direct copy selected. To make a linear transformation, select Linear Transformation or Linear Range Mapping and enter the appropriate data, as explained in Section 5.2.2, *Making transformations* below.

7. Determine which direction you want the bridge to apply.

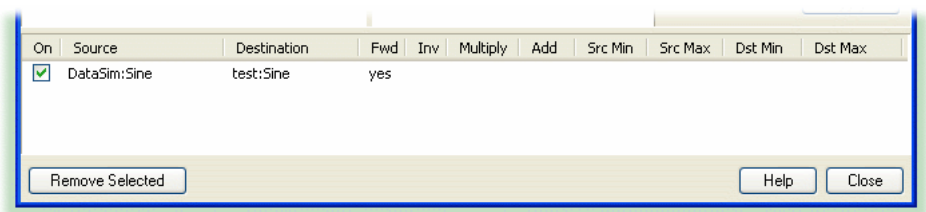


   - Select Forward to change the destination point when the source point changes, but not change the source when the destination changes. If you select Force consistency with this option, and if the destination point gets changed for some reason, then VINE will attempt to force its value to be consistent with the source point value.
   - Select Inverse to change the source point if the destination point changes, but not vice-versa.

     Selecting Inverse will apply the inverse of the transformation, as explained below.

   - Select *both* Forward and Inverse for a bidirectional bridge, where either point changes whenever the other point changes. This combination will deselect Force consistency to eliminate the possibility of conflicting behavior.

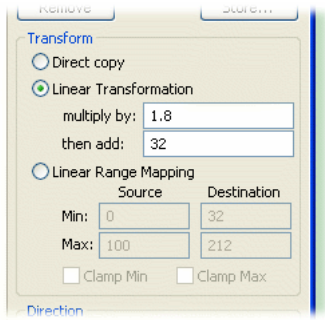
8. Click the Apply button to create and activate the bridge. VINE will create the bridge and update the bridged points immediately.
9. In the bottom panel you can see all the bridges that exist in the system, and the significant information about them.



If you click on a transformation, the source point, destination point, and transform information get displayed in their respective panels. Use the check box at the front of each bridge to activate or deactivate it.

5.2.2. Making transformations

1. Specify the type of transformation by clicking one of the three radio buttons:



   - Direct copy makes no transformations. It just copies the point.
   - Linear Transformation lets you multiply by one value and add another value, such as in the equation `y = mx + b` where the destination point is `y`, the source point is `x`, the multiply by value is `m`, and the then add value is `b`. For example to transform a Celsius source point to a Fahrenheit destination point, you would multiply by `1.8` and add `32`, or

     ```
     Fahrenheit = (1.8 X Celsius) + 32
     ```

     If you have selected the Inverse direction for a transformation, you will get the inverse of the transformation. In this example, you would get a conversion from Fahrenheit to Celsius, or the results of this equation:

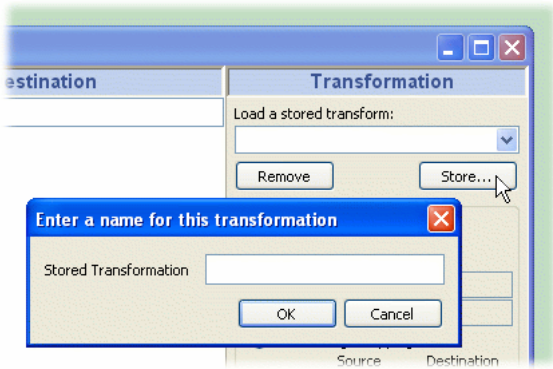
     ```
     Celsius = (Fahrenheit - 32) / 1.8
     ```

     As an alternative to entering transformation values, VINE also offers Linear Range Mapping.

   - Linear Range Mapping lets you enter a range for the source and destination, and VINE automatically calculates the corresponding linear transformation. For example, to create the same Fahrenheit to Celsius transformation, you could use the defaults of `0` and `100` for the Min and Max of the source point. Then you would enter `32` and `212` for the Min and Max of the destination point. As soon as you make these entries, the correct values get entered automatically in the Linear Transformation.

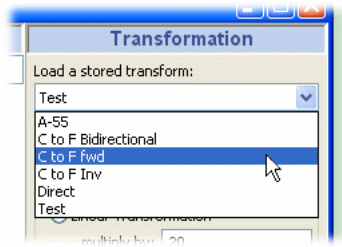
     When you use linear range mapping, you can limit the transformed value to the maximum and minimum by checking the Clamp boxes. The clamps get applied to the point being changed, ie. to the destination point for forward direction, to the source point for inverse direction, and to both points for bidirectional bridges.

2. If you want to save this transformation for future use, click the Store... button at the top of the Transformation panel, and enter a name in the box that pops up.



Once stored, the transformation will become available by name in the drop-down list.

3. To load a transformation, simply select its name from the drop-down list.

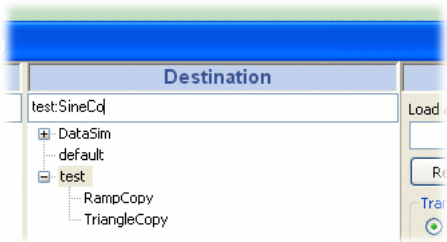


5.3. Creating New Points

A special feature of Skkynet VINE allows you to create new points. Combined with the ability to create new data domains, this lets you create:

- **Personalized data sets** for different users or groups.
- **Aggregated data sets** that combine selected data points from different sources and serve it up from a single server (i.e. VINE).
- **Scaled data sets** that apply one or more transformations to a subset of the data.
- **Temporary data sets** for testing and demonstration purposes.

You can create a new point in the Source or Destination panel of the Bridging Configuration window by typing in a new point name in the entry field.

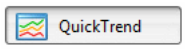


And with a few mouse clicks you can quickly create bridges to new destination points.

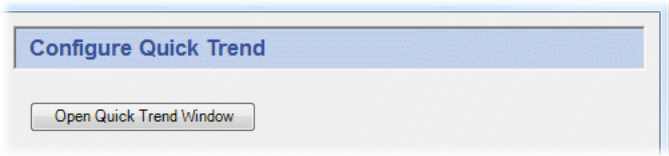
Just click on a source point, click on a data domain, and then click Apply. A destination point with the same name as the source point gets created automatically in the data domain you chose, with the current transformation applied.

You might want to create special data domains for holding sets of destination points. For more information about data domains, please refer to Section 13.4.1, *Data Domains*.

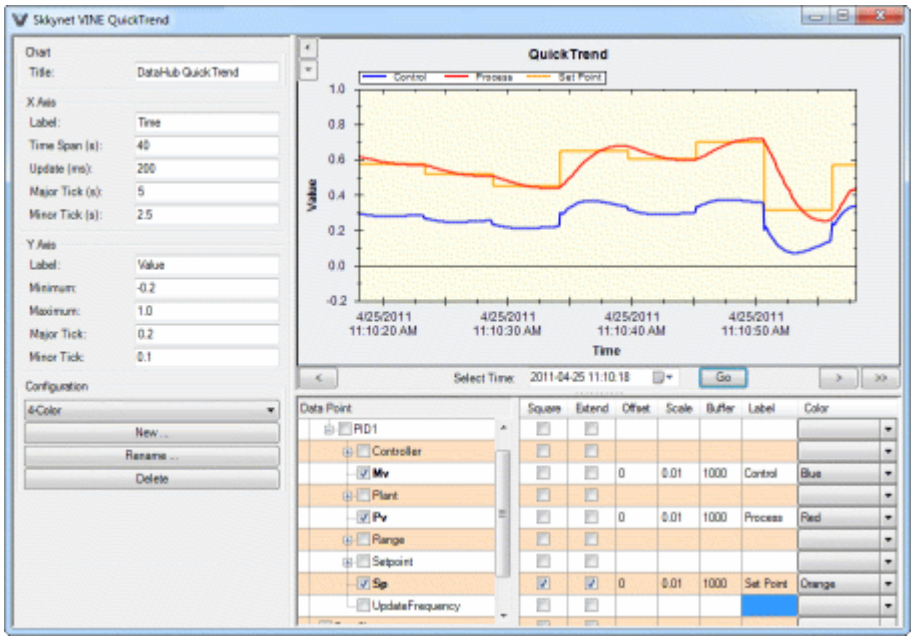
Chapter 6. Using QuickTrend

In the VINE Properties window, select QuickTrend 

.

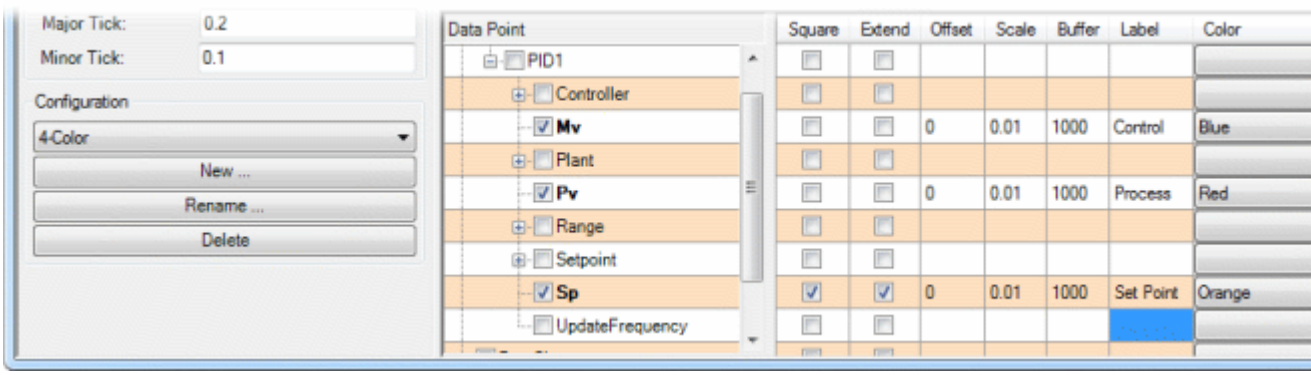


To configure a quick trend, press the Open quick trend window button:



Configuration

QuickTrend supports multiple display configurations, which you can manage using the Configuration options. The top button opens a dropdown list that contains all named configurations. The New..., Rename..., and Delete buttons let you create, rename, and delete configurations.



Data Points

You can select any number of data points to trend, from any data domain. The following options determine how the data will display in the chart.

Square

Removes interpolation of the line between two data changes, giving the plot a step-like appearance. This is useful for square functions.

Extend

When checked, this option extends the plot of a point's value as a straight line until the point changes value. With this option unchecked, no plot is shown until a point changes value. Then a straight line is plotted connecting the original value to the new value.

Offset

A value entered here will be added to each value of the point, creating an offset plot. This lets you view widely differing values in the same window.

Scale

A value entered here will be multiplied by each value of the point, creating an enhanced (or diminished if a fractional value) plot.

Buffer

This value determines how many data changes for this point will be stored in the local history, to allow for scrollbacks to review recently plotted data.

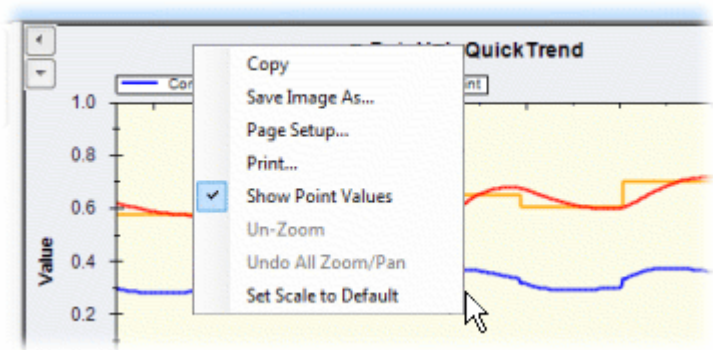
Label

Allows you to change the label for the point, whose default is the simple point name.

Working with the Display

There are a few features of the display that are not immediately obvious.

- You can scroll backwards and forwards through the history of the trend using the left and right arrow buttons, or choose a specific date and time with the calendar and time selector. The double-right arrow button returns the display to real-time trending.
- You can resize the trend display by dragging the gray borders on the left side or bottom. Move the mouse until you see a white, double-headed arrow, and drag.
- To zoom in to a part of the display, drag the mouse over the area that you want to zoom in on. To zoom back out and resume real-time trending, click on the Reset Pan/Zoom button at the bottom of the window.
- For other features, the QuickTrend display has a menu available on a right mouse-click.

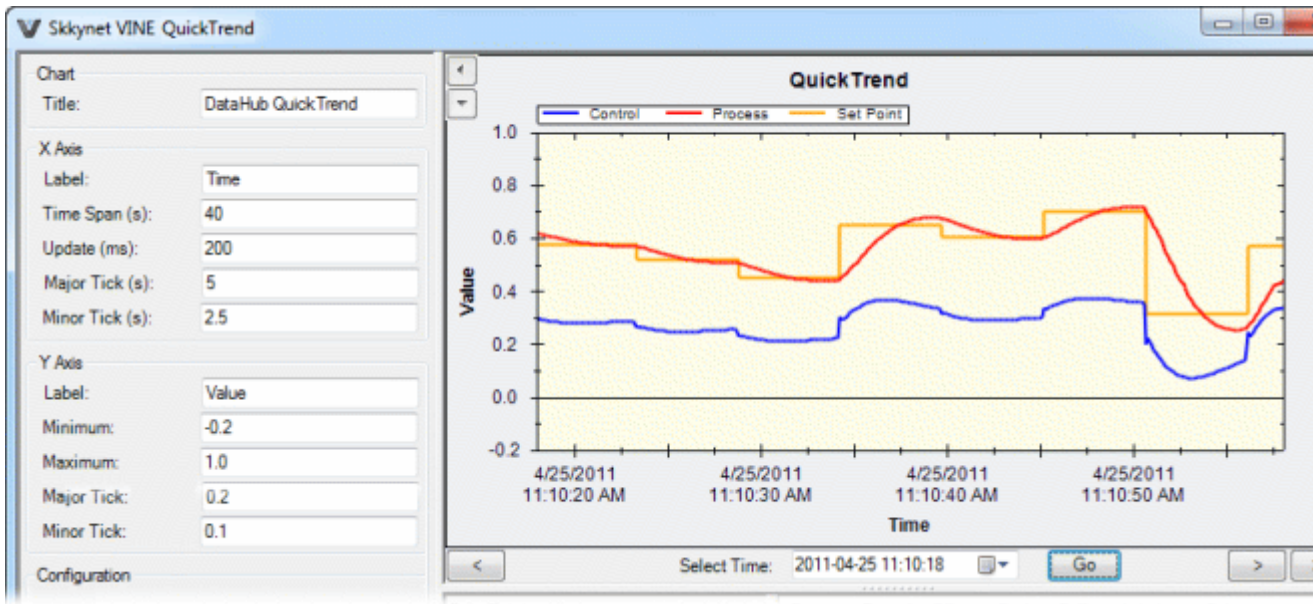


Using this menu you can copy, save, or print the current display, as well as unzoom and undo a zoom or pan. You can also hide or show point values, and reset the scale to the default.

Chart

Title:

The name of the chart, which will appear at the top.



X Axis

The X axis displays the time coordinate.

Label:

Any text string, displays at the bottom of the chart.

Time Span (s):

The total number of seconds that the chart will span.

Update (ms):

The update rate for the trend, in milliseconds.

Major Tick (s):

The time interval between major tick marks, in seconds.

Minor Tick (s):

The time interval between minor tick marks, in seconds.

Y Axis

The Y axis displays the value coordinate.

Label:

Any text string, displays at the far left of the chart.

Minimum:

The minimum value to include in the chart.

Maximum:

The maximum value to include in the chart.

Major Tick:

The value between major tick marks.

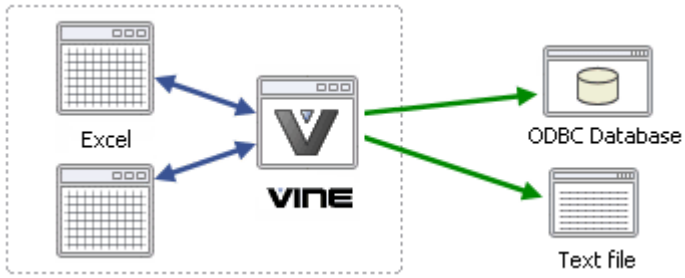
Minor Tick:

The value between minor tick marks.

Chapter 7. Write to a Database

7.1. Introduction

Skkynet VINE can write data to any ODBC compliant database or text file[1].



With this feature of the DataHub you can:

- Log data to any ODBC-compliant database, such as MS Access, MS SQL Server, MySQL, Oracle, and many more.
- Log from any data source connected to VINE.
- Log to existing database tables, or create new tables as necessary.
- Log data to a text file using the `LogFile.g` script.

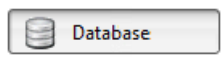
To write data from a database into Skkynet VINE, please refer to Tutorial 3: Writing data from a database to VINE in the VINE ODBC Scripting manual.

Skkynet VINE's ODBC Data Logging interface provides an easy way to connect to a DSN, create or select a table, assign data point properties to table columns, and assign a trigger and conditions for logging points. The fastest way to learn how to use the interface is by watching the web-site video or by using the Quick Start.

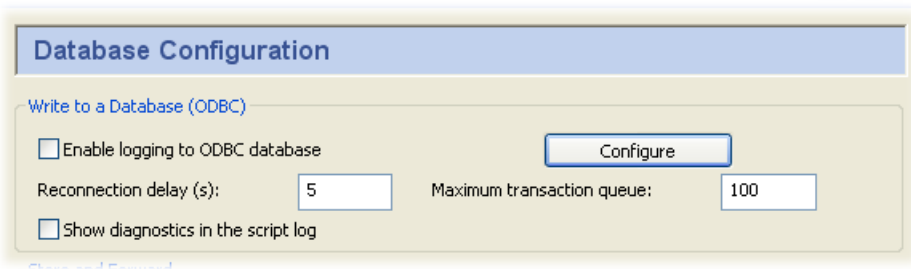
7.2. Quick Start

Here's how to configure Skkynet VINE to write data to a database of your choice. We use a data point from the DataSim program in this example but you can just as easily use your own data point.

Open the ODBC Data Logging window

1. In the Skkynet VINE Properties window, select Data Logging 

.
2. Click the Configure button.

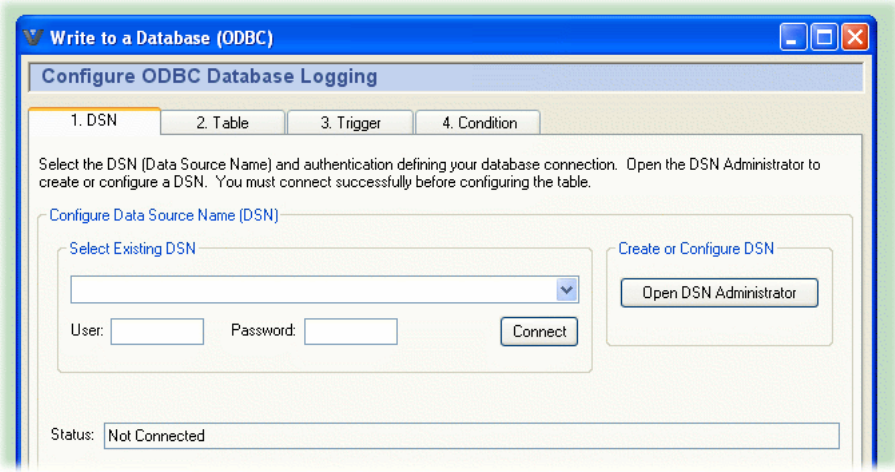


This opens the ODBC Data Logging window, shown below.

The Data Logging Configuration interface is explained in detail in Section 7.3, *Configuring the Queue, Store and Forward*. For now, you can use the defaults.

Connect to the Database

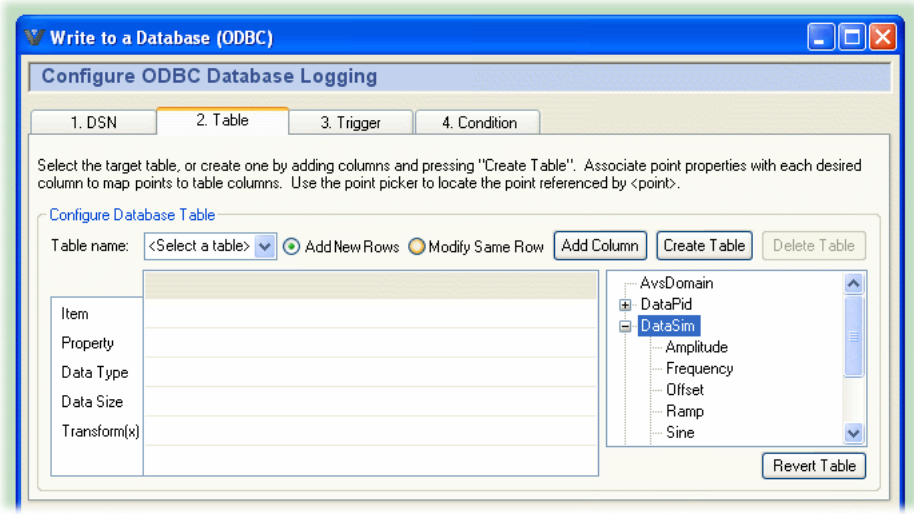
1. Select the 1. DSN tab. A *DSN* is a Data Source Name. Windows uses this name to identify the database you want to connect to.


Write to a Database (ODBC)
Configure ODBC Database Logging
1. DSN
2. Table
3. Trigger
4. Condition
Select the DSN (Data Source Name) and authentication defining your database connection. Open the DSN Administrator to create or configure a DSN. You must connect successfully before configuring the table.
Configure Data Source Name (DSN)
Select Existing DSN
Create or Configure DSN
Open DSN Administrator
User:
Password:
Connect
Status:
Not Connected

2. From the drop-down box, select a DSN. If you do not have any DSNs, or you wish to create a new DSN, you can do this by opening the DSN Administrator. Please refer to Setting up a DSN for more details.
3. Enter the appropriate user name and password (if required), and click the Connect button. A "`Connected to ...`" message should appear in the message box. If you get an error message in the box, consult your system administrator.

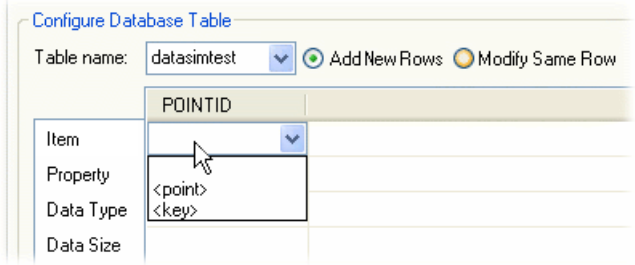
Configure a Table

1. Select the 2. Table tab.



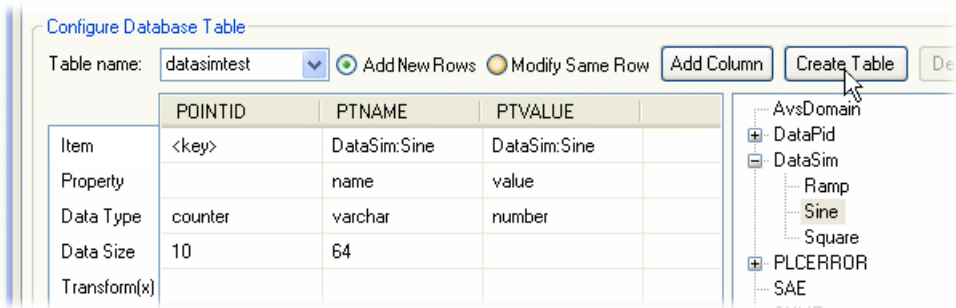
2. Start the DataSim program if it isn't already running, and ensure that it is connected to VINE.
3. In the Table name field type: `datasimtest`.
4. Click the Add Column button, type: `POINTID` in the pop-up dialog, and then click OK.

5. Click under the POINTID label in the Item row.



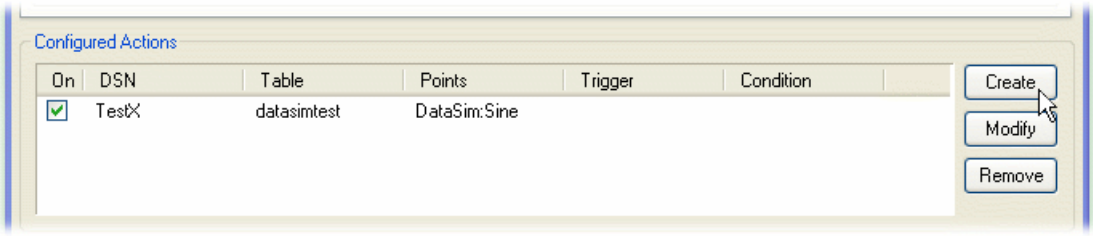
6. Select <key> from the drop-down list. Notice after you make your selection that the word `counter` gets entered automatically for the Data Type.
7. Click the Add Column button and enter the name PTNAME.
8. In the point-picker list on the right, expand the `DataSim` data domain and select the point named `Sine`.
9. In the PTNAME column, click in the Item row and select `<point>`. The full name of the point, `DataSim:Sine`, should appear.
   for Property in that column select `name`.
   for Data Type select `varchar` (or the equivalent).
   for Data Size the system might have entered a value for you. If not, type in a value like 64 and click **Enter**.
10. Click the Add Column button again and add the column name PTVALUE. Then make these entries:
    for Item select `<point>`. (The point name, `DataSim:Sine`, should appear.)
    for Property select `value`.
    for Data Type select `number` (or the equivalent).
    for Data Size, depending on your database, you might not be able to enter anything. If you are able to make an entry,

    The entry fields should now look similar to this:



11. Click the Create Table button. If successful, you have now created a new table in the database specified by your DSN. You can open your database program and view it to verify. If you get an error message, check your entries above carefully to ensure they are compatible with your database. For example, some databases will not allow spaces or special characters in table and column names.

    Once the table is created, you cannot add any more columns. However, you can delete the table using the Delete Table button. This will delete the table from the database, but all of your entries will remain in the entry fields. You can then add more columns if you wish, and recreate the table. You can easily rename, insert, or delete a column by right-clicking on the column name for a pop-up menu. For more information about creating and modifying tables, please refer to Section 7.5, *Configuring a Database Table*.
12. When you have the table the way you want it, go down to the Configured Actions box and click the Create button.



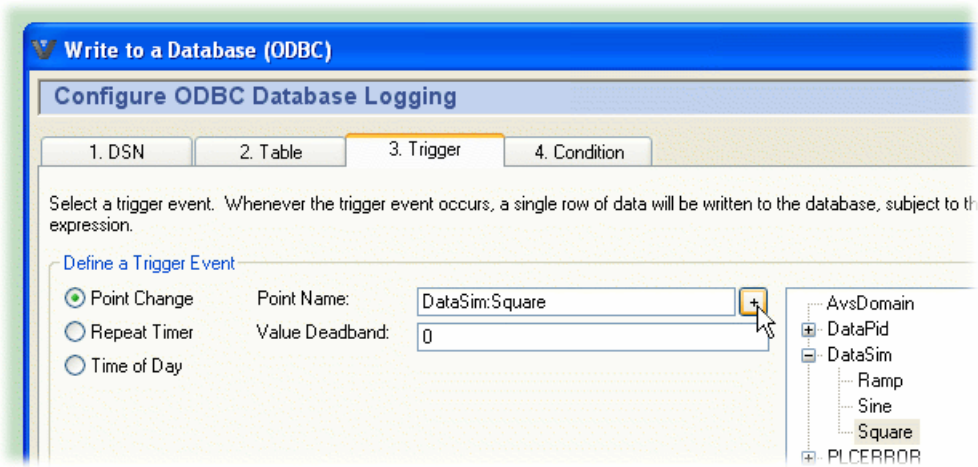
A new configured action should appear in the list. For more information about configured actions, please refer to Section 7.9, *Configured Actions*

Next, to get VINE to write the data, you need to assign a trigger.

Assign a Trigger

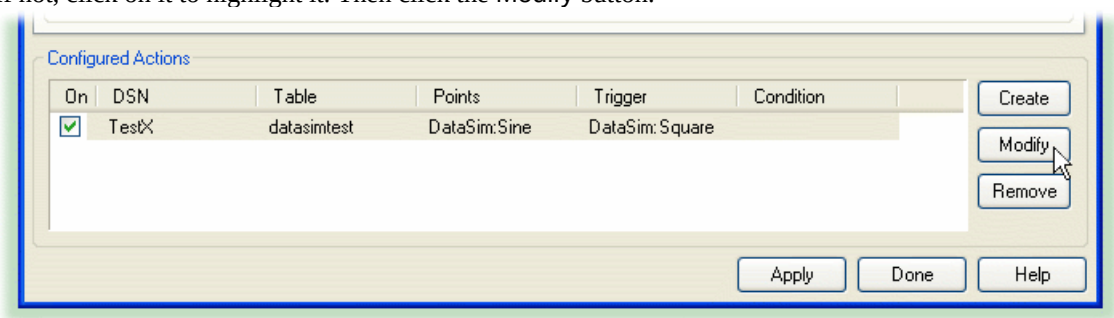
For this example, we will trigger the action whenever the `DataSim:Square` point changes value.

1. Select the 3. Trigger tab.
2. From the point selector, expand the `DataSim` data domain and select the point `Square`.
3. Click the + button to the right of the Point Name field. The point name `DataSim:Square` should fill in for you.



You can choose any point for the trigger, including the point that gets written, such as `DataSim:Sine` in our example. For more information about triggers, please refer to Section 7.7, *Assigning a Trigger*.

4. In the Configured Actions box, make sure that the configured action you just created is highlighted. If not, click on it to highlight it. Then click the Modify button.



Your configured action should now display the `DataSim:Square` point as your Trigger.

5. Click the Apply button to activate the configured action.

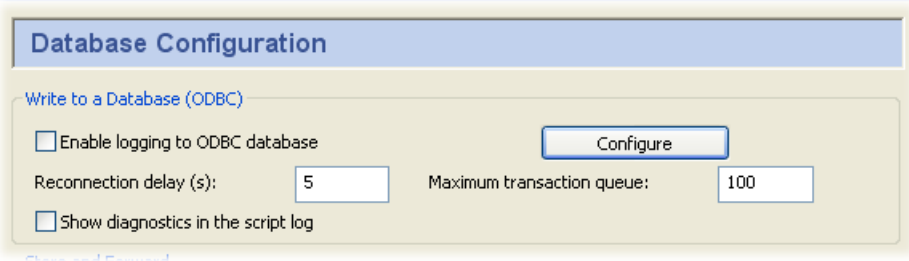
6. Open the Script Log to check for error messages and ensure that your data is being written successfully. Each write action to the database gets logged here. You can also verify the writes by querying the database itself.

You have just configured an action that logs the name and value of the `Sine` point in the `DataSim` data domain whenever the value of the `Square` point changes. Now you can create tables to log your own data, with triggers based on points or timers.

The remaining sections in this chapter explain the interface in more detail, and introduce the option of setting specific conditions for logging, if desired.

7.3. Configuring the Queue, Store and Forward

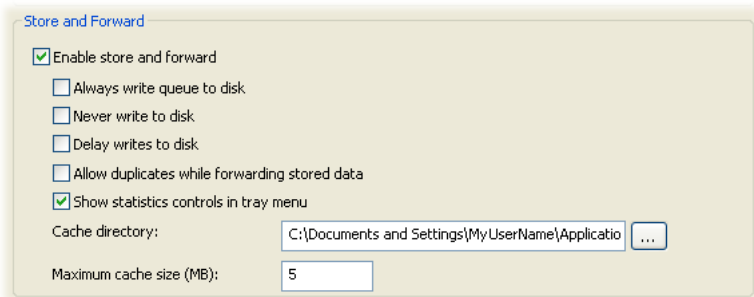
VINE maintains an in-memory *transaction queue* of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. The Maximum transaction queue option lets you modify the depth of this queue, with a default of 100 messages. The Reconnection delay (s): option specifies the number of seconds before a reconnect is attempted if the ODBC connection is broken. And the Show diagnostics in the Script Log option lets you view messages about the connection in the Script Log.



Store and Forward

The term *store and forward* refers to a type of database connection where the data is stored locally to disk and then later forwarded to the database. Skkynet VINE performs an advanced form of store and forward that only writes to disk if the database is not connected, or has been paused. If the database is available, the data will be transmitted directly to the database. This means that there is no penalty for using store and forward during normal operation. The VINE store and forward mechanism uses two levels of disk caching to ensure that all data gets logged, and nothing is lost.

When the database first becomes available after an outage, VINE starts writing cached values to the database, and continues writing new values to the cache. In this way, the values are inserted into the database in the order in which they are generated by the system. Once the cache is cleared, VINE then starts writing new values directly to the database.



Enable store and forward

Activates the store and forwarding feature.

Always write queue to disk

Data in the transaction queue will be written to disk cache first, and from there to the database. The safest protection against a crash is to check this box, and uncheck Delay writes to disk (below).

Never write queue to disk

The data in the transaction queue will be only stored in memory, and never written to disk.

Delay writes to disk

Data in the transaction queue will be written to disk at the most opportune times. The safest protection against a crash is to uncheck this box, and check Always write queue to disk (above).

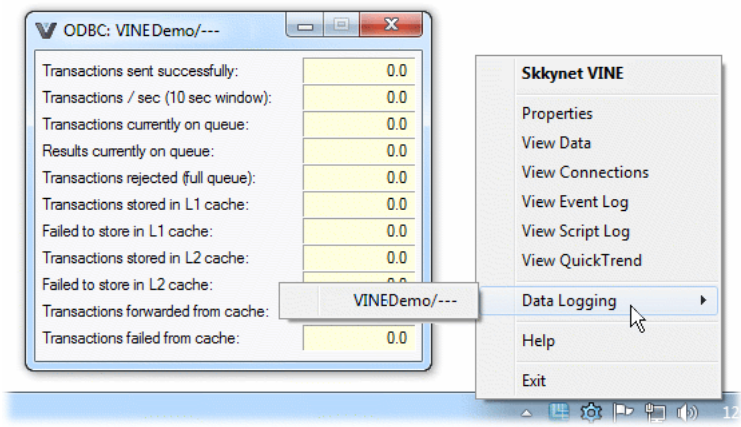
Allow duplicates while forwarding stored data

If the network breaks while transmitting data from a cache, Skkynet VINE needs to know how to handle any already-sent data when it reconnects. Leaving this box unchecked will require Skkynet VINE to track its cache position at all times, and modify that information each time a value is sent. This will impact the speed of every transmission, but it will ensure that no values get transmitted twice.

Checking this box will cause VINE to simply start from the beginning of the queue or cache on each reconnect, and retransmit some data. This significantly reduces data-handling complexity and decreases transmission rates. This option is particularly useful if network breaks are frequent and some duplication of logged data is acceptable.

Show statistics in tray menu

Adds a Data Logging entry to the VINE's system tray menu, which lets you open a statistics window:



Transactions sent successfully:

The number of transactions that were sent, either directly to the database, or to the disk cache.

Transactions / sec (10 sec window):

The sending rate for transactions, calculated over the past 10 seconds.

Transactions currently on queue:

The number of transactions in the queue.

Results currently on queue:

Not yet documented.

Transactions rejected (full queue)

The number of transactions that were rejected from the queue because it was full.

Transactions stored in L1 cache

The number of transactions taken off the queue and put into the first-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L1 cache

The number of transactions that were not able to be stored in the first-level cache.

Transactions stored in L2 cache

The number of transactions taken off the queue and put into the second-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L2 cache

The number of transactions that were not able to be stored in the second-level cache.

Transactions forwarded from cache

The total number of transactions forwarded from both caches. This number should be the sum of L1 and L2, once all transactions have been forwarded, and as long as VINE was started up with no cache on disk.

Transactions failed from cache

The number of transactions attempted from cache, could not be successfully delivered, and were stored for later transmission. This phenomenon may occur the first time that VINE learns that the database is not available. For example, you'll see this for every network break if you've checked Always write queue to disk.

Cache directory:

The path and directory name for the cache.

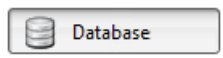
Maximum cache size (MB):

The amount of disk space to allocate for the cache, in megabytes.

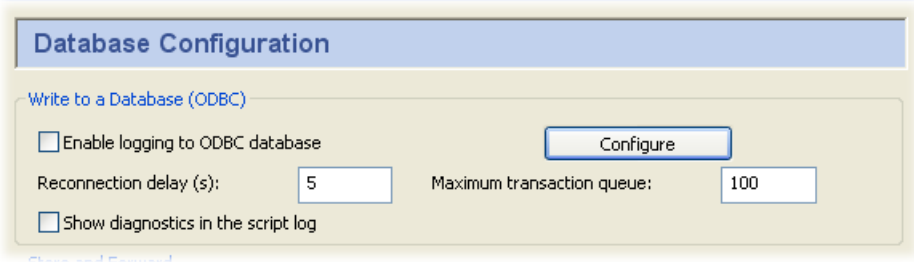
7.4. Setting up the DSN (Data Source Name)

A *DSN* is a Data Source Name. Windows uses this name to identify the database you want to connect to. This tab lets you select an existing DSN, or create a new one if necessary.

Open the ODBC Data Logging window

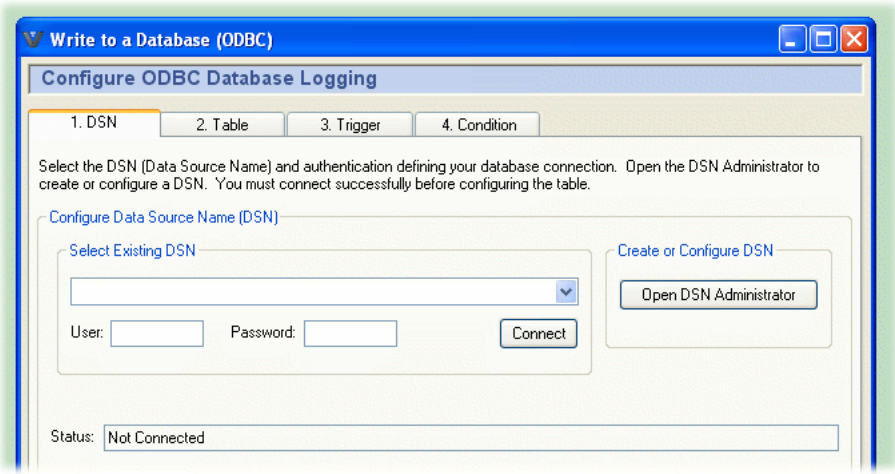
1. In the Skkynet VINE Properties window, select Data Logging 

.
2. In the Configure Database (ODBC) Data Logging section click the Configure button.



The Data Logging Configuration interface is explained in detail in Section 7.3, *Configuring the Queue, Store and Forward*.

3. Configure your DSN as explained below.


Write to a Database (ODBC)
Configure ODBC Database Logging
1. DSN
2. Table
3. Trigger
4. Condition
Select the DSN (Data Source Name) and authentication defining your database connection. Open the DSN Administrator to create or configure a DSN. You must connect successfully before configuring the table.
Configure Data Source Name (DSN)
Select Existing DSN
Create or Configure DSN
Open DSN Administrator
User:
Password:
Connect
Status:
Not Connected

Selecting a DSN

1. To select a DSN, choose one from the drop-down box, and then enter the appropriate user name and password.

If your DSN is using Windows Authentication, leave the User and Password fields blank.

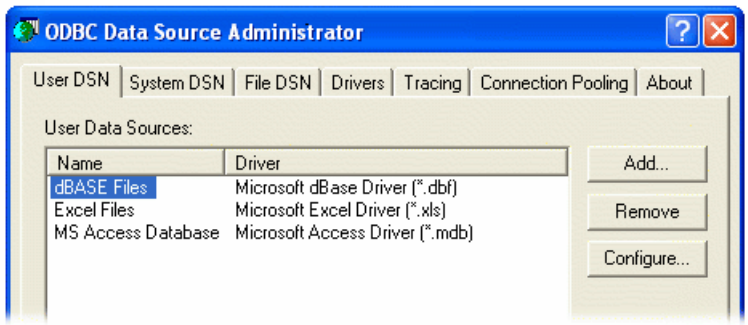
2. Click the Connect button. A "`Connected to ...`" message should appear in the message box. If you get an error message in the box, consult your system administrator.

Creating or Configuring a DSN

Windows has different configuration programs for 32-bit and 64-bit DSNs. Skkynet VINE uses 32-bit DSNs. To configure a 32-bit DSN in a 64-bit system, you need to run:

```
c:\windows\syswow64\odbcad32.exe
```

1. To create or configure a DSN, click the Open DSN Administrator button. This opens the ODBC Data Source Administrator window.



2. Select the User DSN or System DSN tab, depending on how you plan to access your database.

   A user DSN is only available to the current user account, while a system DSN is available to any user account on the computer.

3. Now you can add a new database or configure an existing one.

   Add a new database

   1. Click the Add button. The Create New Data Source window will open, displaying a list of data source drivers.
   2. Select the data source driver that corresponds to your ODBC database. A data source setup window will open. Each data source setup window is different, but you should be able to find the appropriate entry fields easily enough.
   3. Enter the data source name and select the database.
   4. Enter any other required or optional information such as login name, password, etc. What entries need to be made and where they are entered depends on the particular data source setup window you are using.
   5. Click OK to return to the ODBC Data Source Administrator window. You should be able to see the new database and driver listed. If you need to make any changes, you can configure an exiting database, as explained below.

   Configure an existing database

   1. Select a data source name and click the Configure... button. This takes you to the data source setup window (explained above) where you can make changes to the configuration.
   2. Make your changes and click OK to return to the ODBC Data Source Administrator window. Any time you need to make a change, you can go to this window.

4. When you are satisfied everything is correct, click the OK button to exit the ODBC Data Source Administrator.

Once you have created a DSN, you can select it as explained above.

7.5. Configuring a Database Table

After you have set up a DSN you can create a table or select an existing one, and then assign points and properties to the columns of the table.

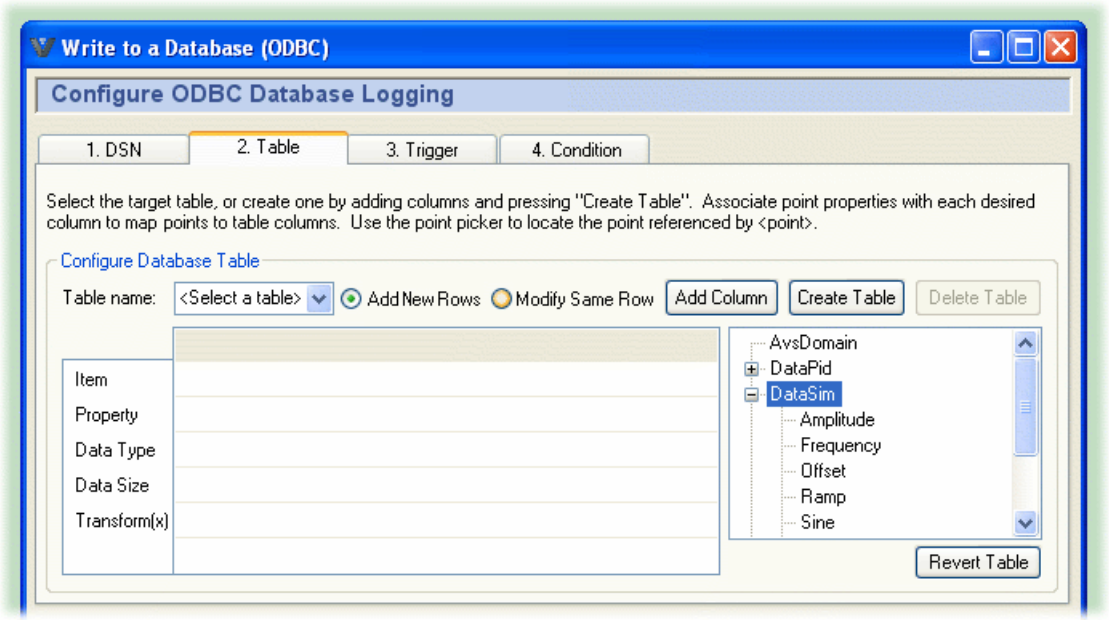


Table Selection or Creation

You need to either select an existing table, or create a new one.

- **Selecting a table** allows you to use a table that you have created in your database program. When you select an existing table, you cannot add columns or change column names. To select a table, choose a table name from the drop-down list.
- **Creating a table** provides a way to design and create a table from within this interface.

1. From the Table Name drop-down list, select `<Create a new table>`.
2. In the dialog box, type in a table name and click OK.

   Now you will have to add at least one column.
3. Click the Add Column button to create a column.
4. In the dialog box, type in a column name and click OK.

   At this point you can add more columns, or you can assign points (as explained below). You can easily rename, insert, or delete a column by right-clicking on the column name and selecting Rename Column, Insert Column, or Delete Column from the pop-up menu.
5. When all of your columns have been created and named, you can create the table in the database. Once created, you cannot add columns to the table. To create the table, click the Create Table button.

   If, after creating the table, you need to make changes to it, and it has no data in it that you need, you can click the Delete Table button. This will delete the table from the database but will leave all of the configuration you've done intact in this window. You can then add more columns or make changes to points and properties and recreate the table. This can be done as often as you need to.

Add New Rows or Modify the Same Row

For any given table, whether existing or newly-created, you will need to decide whether you are going to add new rows, or modify the same row with new data each time it changes.

- **Add New Rows** creates a new row in the table each time data is logged.
- **Modify Same Row** writes to the same row in the table each time data is logged.

Assigning a Key

You have the option to make one column, often the first column in a table, a key column.

- **An auto-incrementing key** is commonly configured for Add New Rows. A key column is optional for adding new rows, but if you choose to have one, it **must** be auto-incrementing. Please refer to Section 7.6, *Key Columns* for more details. An auto-incrementing key can be configured as follows:

  Item: Choose `<key>` from the drop-down list.
  Property: (none)
  Data Type: The appropriate auto-incrementing integer type for your database.

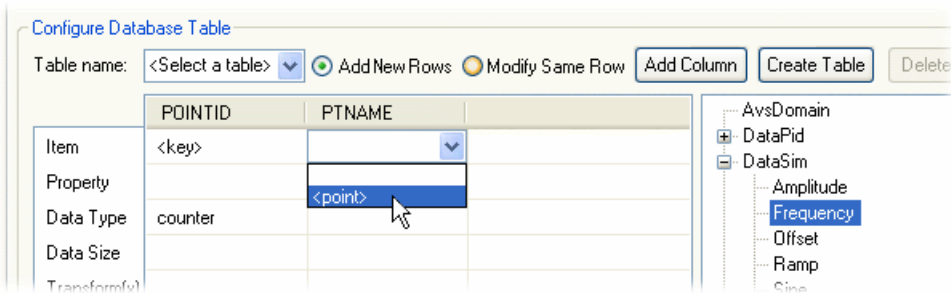
- **A single key column** is required for Modify Same Row. VINE uses the key value to determine which row to modify. Please refer to Section 7.6, *Key Columns* for more details. The key can be configured as follows:

  Item: Type in a VINE point name, or any other value.
  Property: `<key>`
  Data Type: The appropriate type for your database (typically VARCHAR, NVARCHAR or CHAR).

Due to a Windows bug, in the first column the interface won't respond unless you click directly below the text of the column name. We are working to resolve this.

Assigning points and properties

When you have selected a table, or you have at least one column in a table you are creating, you can assign points and their properties to the various columns.



Item

First, choose a point from the point-picker list on the right. Then click in the Item row and select `<point>` from the drop-down list. Optionally, you can type in the name of the point. Leaving the Item blank allows you to choose the Property of `clock` to display the system time, or `clockms` to display the number of milliseconds after the second of the system time.

Property

Select which property of the point you want written to the database in this column:

`name`

The name of a the point shown in the Item field.

`value`

The value of the point shown in the Item field.

`quality`

The quality of the point shown in the Item field.

`timestamp`

The time stamp of the point shown in the Item field. This will include the milliseconds, but many databases, such as MS Access, ignore the milliseconds and store only the seconds. Other databases such as MySQL and MS SQL Server include the milliseconds in a time stamp. For example:

- Databases like MySQL and MS SQL Server:

	Column A
Enter for Property	`timestamp`
Enter for Data Type	`datetime` or `timestamp` or `date`

- Databases like MS Access:

	Column A	Column B
Enter for Property	`timestamp`	`timems`
Enter for Data Type	`datetime`	`number` or `integer`

`timems`

The millisecond component of the `timestamp`, generally used in conjunction with `timestamp`. You only need this if your database cannot store the millisecond component of `timestamp`.

`timestampUTC`

The same as the `timestamp`, but in UTC time. You can use `timems` in conjunction with this as well.

`clock`

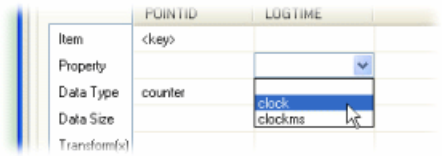
The current system time. This will include the milliseconds, but like `timestamp` (above) many databases ignore the milliseconds and store only the seconds.

`clockms`

The millisecond component of `clock`, generally used in conjunction with `clock`. Like `timems` (above), you only need this if your database cannot store the millisecond component of `clock`.

Using `clock` and `clockms` - some tips:

- You must leave the `Item` field blank to select either of these options.



- **Example:** If the time is `12:34:56.789`, `clock` will be written as `12:34:56.789` in databases that accept milliseconds, and as `12:34:56` in databases that do not. A `clockms` property will be written as `789` in all databases.
- The `clock` and `clockms` properties allow you to log the system time as a column in the table, so that your record can contain the system time along with a number of different point values, for example:

```
System Time      Point1      Point2      Point3
08:12:56.000     43.883       3.727     213.905
```

Data Type

The data type that the database should associate with the property.

Data Size

In some cases this is entered automatically, in other cases it is not used, but sometimes it is possible or necessary to enter a size, such as the number of characters in a text string, or the number of bytes.

Transform(x)

This allows you to modify the entry or to insert a text string. For example:

- `(x * 100) + 25` could be used to multiply a point value by 100 and add 25 to the result.
- `"Tank Level"` would insert the string `Tank Level` instead of, say, the point name.

7.6. Key Columns

A database table can have a special column (or set of columns) that is designated as a *key*. Every value in the key column(s) is unique. By designating a key in a table, you are providing the database engine with a guarantee that any search on the key will produce either zero or one row. Effectively, when a search matches a value in the key column, the row containing that key is guaranteed to be unique in the table.

A database table does not require a key. The key is effectively a hint to the database engine to improve efficiency and to guarantee uniqueness. You can still search and modify a table that has no key.

When working with the VINE Data Logging interface, you have the option of specifying a key column. The interface will only allow single-column keys. You will not be able to specify more than one key column, and if you choose an existing table with multiple key columns, the interface will fail.

Your choice of key will depend on whether you have chosen Add New Rows or Modify Same Row for your database table.

Adding New Rows

If you choose to add (insert) new rows into your database table, you do not require a key column. Each new logging event will create a new row and populate it with the data that you have configured. Since

you are continually adding rows to the table, there is no data that can be guaranteed to be unique within a column. Any one of point name, quality, value or timestamp may be repeated.

Consequently, if you do wish to have a key column in your table, it must have one important property: it must be automatically generated and guaranteed to be unique. The key value cannot be derived from a VINE point. Most database engines support the concept of a *counter* or an *auto-increment* numeric value. If you choose to use a table containing a key, the key column must be of the appropriate auto-incrementing integer type for your database.

You can designate a key by choosing the `<key>` option for the Item entry of the database table when configuring a table through the Data Logging interface. If you are using an existing table that already contains a key, you must specifiy in the Data Logging interface that the column is a key column.

Modifying the Same Row

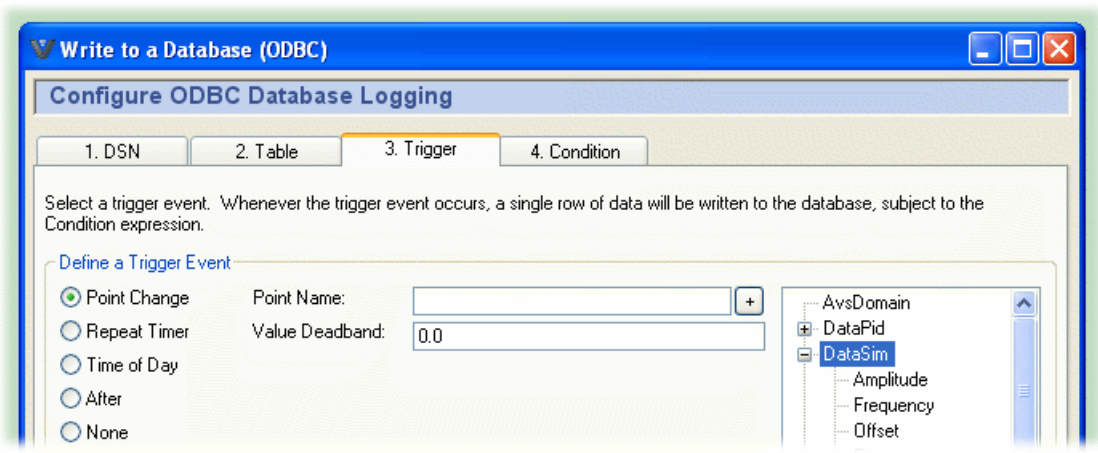
If you choose to modify a row in the database, such that any new data will overwrite the existing data in the row, you must be able to uniquely identify that row. This means that you **must** have a key column.

The key column can be any type, and does not need to be auto-incrementing. Since the row is overwritten whenever new data is available, no new key value is generated. It is common to supply the VINE point name as a key, with the key field defined as a string type (typically VARCHAR, NVARCHAR or CHAR). You can do this by selecting a point in the point picker tree, then choosing the `<point>`; option for the Item entry of the database column, and `<key>` for the Property field.

If you wish to specify a key value other than a point name, type your own value into the Item entry instead of using the point name. You must then choose `<key>` for the Property entry.

7.7. Assigning a Trigger

A trigger is an event that causes a row of data to be written to your database table. A trigger event can be either a point value change, a timer event, or a calendar event. You can assign a different trigger for each row, or an identical trigger to any number of rows. An action can be configured to execute on every trigger event, or you can assign trigger conditions that are evaluated whenever a trigger occurs, to determine if the action should be executed.



The three kinds of triggers are:

- **Point Change** fires whenever a specified trigger point changes.

1. Type the name of the point into the Point Name box, or select the point using the data tree on the right, then click the + button.
2. (Optional) Enter a value deadband if you want to filter out extraneous data. The number you enter will specify a high and low (plus or minus) range. Any value change falling within that range will not cause the trigger to fire. A positive or negative change greater than this value will activate the trigger and cause the row to be written.

To create a trigger that gets reset automatically, please refer to An Auto-Resetting Trigger in Section 7.8, *Setting Trigger Conditions*.

- **Repeat Timer** fires cyclically, each time the number of seconds elapses.
- **Time of Day** fires at the time you specify. You can enter:
  - A number, indicating a specific value. For example, a `0` in the seconds field would cause the event only on the 0th second of the minute. A `30` would indicate only on the 30th second of the minute.
  - A list of numbers, separated by commas. For example, entering `0,15,30,45` in the minutes field would indicate that the event should fire on the hour and at 15, 30 and 45 minutes past the hour.
  - A range of numbers, separated by a dash. For example, entering `8-18` in the hours field would indicate that the event should fire every hour from 8 a.m. to 6 p.m.. Ranges can be mingled with lists, as in `0,4,8-16,20`.
  - An asterisk (`*`) indicates that the event should fire for every possible value of the field. For example, a `*` in the seconds field would cause the event to fire every second. A `*` in the hours field would cause the event to fire every hour.

To regularly log a record on specific days of the week, please refer to Section 7.8, *Setting Trigger Conditions*.

The ranges of the fields are:

Year:	`1970-*`	Hour:	`0-23`
Month:	`1-12`	Minute:	`0-59`
Day:	`1-31`	Second:	`0-59`

The year and month are entered differently here than for the Gamma `localtime` function, as explained in Time Conditions.

Examples:

- These entries:

would cause a row to be logged at 8:45 every day, every month, and every year.

- These entries:

would cause a row to be logged every hour on the 15th day of each month, every year.

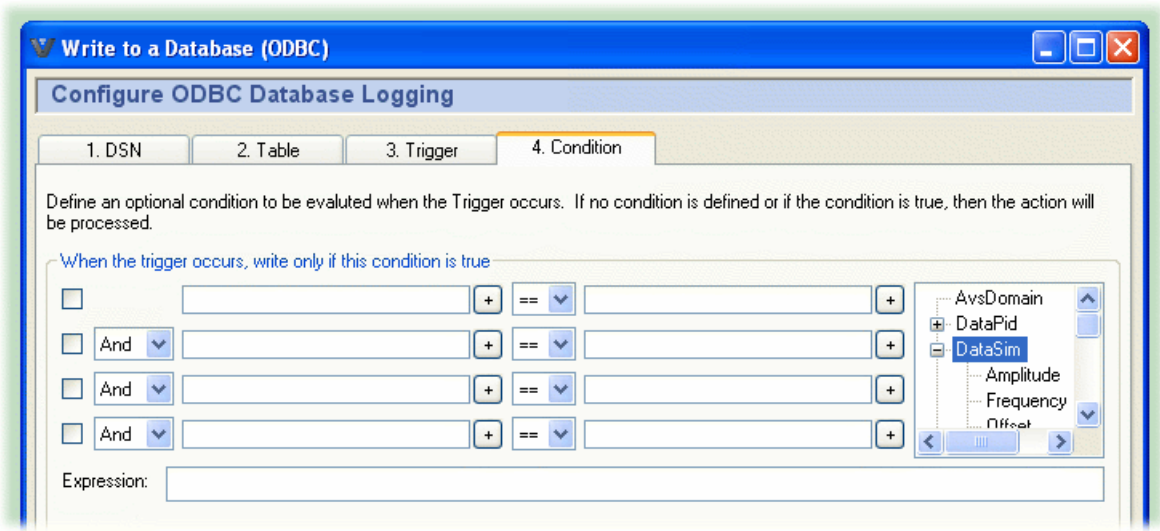
- These entries:

would cause the event to fire every second for 5 minutes, every two hours between 8 a.m. and 6 p.m.

- **After** fires once, when the specified number of seconds elapses.
- **None** configures no trigger.

7.8. Setting Trigger Conditions

Each logging action can have up to four conditions that determine whether a row gets written to the database when the trigger fires.



Fill in the conditions according to the guidelines below. Check the box next to the condition to apply it. As you make entries, the corresponding *Gamma* code will appear in the display. Gamma is VINE's built-in scripting language. The code that appears in the Expression box is the actual code that gets run by Gamma. The order of precedence for "`And`" and "`Or`" operators (`&&` and `||`) is first `And`, then `Or`.

Point Value Conditions

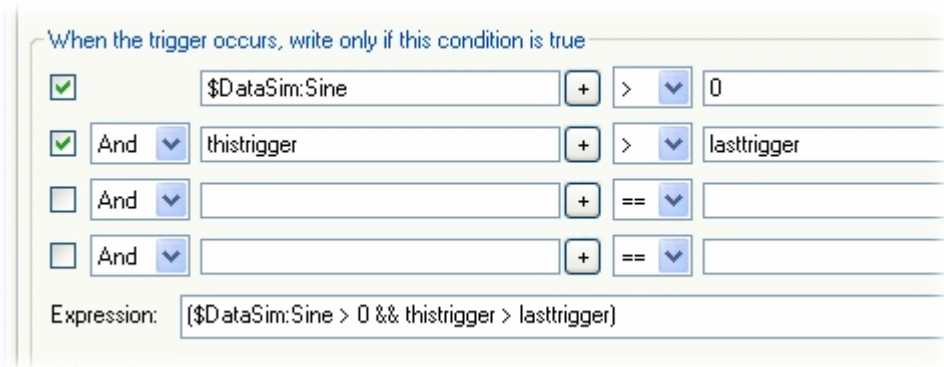
Point names can be entered on either or both sides of the comparison. They can be picked from the data tree list, or typed in. Each point name needs to have a dollar sign ($) in front of it to indicate to Gamma that this is a VINE point. You can put numerical values into either side of the comparison.

When you enter a point name in a condition field, the current value of the point will be used in the evaluation. For example, you could define a condition that states that whenever the trigger event occurs, the action will only be executed if another point value is within a certain range.

There are some automatic variables available for working with point values:

- `lasttrigger` - the value of the trigger point the last time this trigger was fired (even if the condition failed).
- `thistrigger` - the value of the trigger point now (even if the condition failed).
- `lastevent` - the value of the trigger the last time the event was actually executed (the condition succeeded).
- `this` - the trigger point itself, not the value of the point.

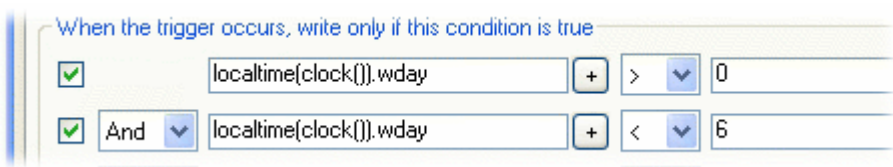
You can use these variables in any condition that is triggered by a data value change. For example, you could create some conditions like this:



Time Conditions

This provides an additional way to restrict the time, day, month, etc. when a message gets sent. In addition to the options on the triggers, here you have day-of-week condition statements which can give you more flexibility for evemts based on specific days of the week. These will work with any type of trigger event.

You can use the Gamma functions `clock` and `localtime` to specify particular days of the week. For example, these entries:



would create this Gamma code::

```
(localtime(clock()).wday > 0 && localtime(clock()).wday < 6)
```

which would cause data to be logged only Monday through Friday. The function `localtime` returns a class whose members contain information about the date, as follows:

`.sec`	The number of seconds after the minute (`0` - `59`).
`.min`	The number of minutes after the hour (`0` - `59`).
`.hour`	The number of hours past midnight (`0` - `23`).
`.mday`	The day of the month (`1` - `31`).
`.mon`	The number of months since January (`0` - `11`)
`.year`	The number of years since 1900.
`.wday`	The number of days since Sunday (`0` - `6`).
`.yday`	The number of days since January 1 (`0` - `365`)
`.isdst`	`1` if daylight saving time is in effect, `0` if not, and a negative number if the information is not available.

The year and month are entered differently here than for Time of Day trigger conditions, as explained in Section 7.7, *Assigning a Trigger*.

There are two automatic variables available for working with time values:

- `lasteventtime` - the system clock time (UTC floating point) that the last event was executed.
- `curtime` - the system clock time (UTC floating point) when the current trigger occurred.

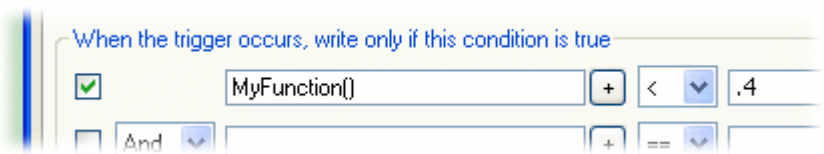
Custom Conditions

If the conditions you need to meet are beyond the scope of this interface, you can use a Gamma function to express virtually any condition you need. Then you can insert the function into one of the condition boxes, and set a condition based on the return value of the function.

To do this is you can create a VINE script (`.g` file) that contains only the functions you will be using for conditions, without any classes or methods. For example, here is the complete contents of such a file, named `MyConditions.g`:

```
function MyFunction ()
{
    myvalue = $DataSim:Sine;
    princ("Value when the trigger fired: ", myvalue, "\n");
    myvalue;
}
```

This function prints the value of the `DataSim:Sine` point, and returns its value. We can use this function as a condition by calling it from one of the condition boxes in the interface, like this:



When the trigger fires, `MyFunction` is called, and the return value gets checked to see if it is less than `.4`. If so, the data gets logged.

Below are two practical examples demonstrating custom functions. The first creates an automatically resetting trigger, and the second lets you test for changed values before logging a point, which is useful if you are logging based on a timer.

An Auto-Resetting Trigger

This script can turn any VINE point into a trigger that automatically resets. To use it, you first need to load and run the `TriggerFunctions.g` script (shown below and included in the installation archive). Then, if you put this formula:

```
HighWithReset($TriggerPoint)!= nil
```

into the condition boxes, whenever the *`TriggerPoint`* changes to a non-zero number in VINE, your trigger will fire. The script waits for a millisecond, then resets the *`TriggerPoint`* back to zero. The second function works similarly, but triggers on a change to zero, instead of a change to a non-zero number.

TriggerFunctions.g

```
/*
 * This file contains handy functions to perform more complex condition
 * handling in the Condition tab of the data logging and email interfaces.
 */

/*
 * Test a trigger point for a non-zero value.  If the point is non-zero,
 * create a delayed event to reset the point to zero, and return true,
 * indicating that the condition has succeeded and the action should proceed.
 * If the value is 0, then simply return nil indicating that the action
 * should not proceed.  We need to test for zero because when we reset the
 * trigger point to zero a second data change event will occur.
 *
 * The argument is unevaluated, so the condition should look like this:
 *    HighWithReset($default:TriggerPoint) != nil
 */

function HighWithReset(!triggerpoint)
{
    local   value;
    if (!undefined_p(value = eval(triggerpoint)) && value != 0)
    {
        after(0.001, 'setq(@triggerpoint, 0));
        t;
    }
    else
    {
        nil;
    }
}

/*
 * This is the inverse of HighWithReset (see above).  If the trigger point
 * is zero, perform the action and set the trigger point to 1.  If the
 * trigger point is non-zero do nothing and return nil.
 */
function LowWithSet(!triggerpoint)
{
    local   value;
    if (!undefined_p(value = eval(triggerpoint)) && value == 0)
    {
        after(0.001, 'setq(@triggerpoint, 1));
        t;
    }
    else
    {
        nil;
    }
}
```

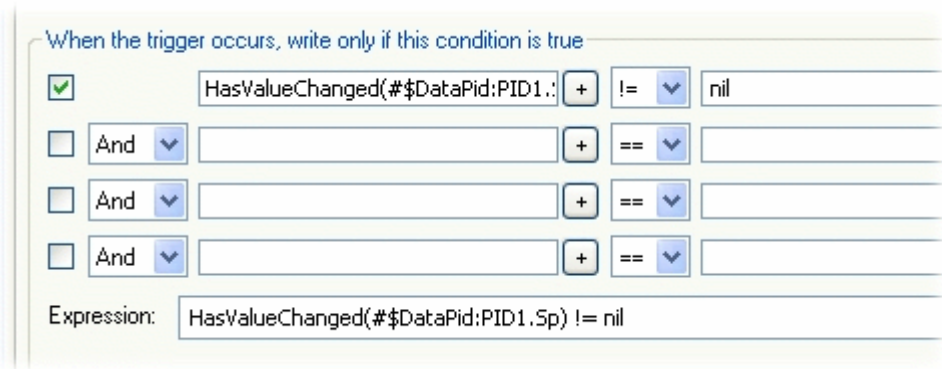
Test for Changed Values Before Logging

If you are logging data based on a timer, you can use this script to check the previously logged value, and only log a new row in the database if the value has changed. To use the script, you first need to load and run the `LogFunctions.g` script (shown below and included in the installation archive). Then, if you put this formula:

```
HasValueChanged(#$TestPoint)!= nil
```

into the condition boxes, when the *`TestPoint`* has not changed, the data for that row will not be logged.

The call to `HasValueChanged` uses the syntax `#$`*`pointname`* to specify the point. If you select the point from the tree to the right and press the + button, it will not include the # symbol. You need to manually add it. You can see the complete condition expression in the Expression: line beneath the condition selectors, like this:



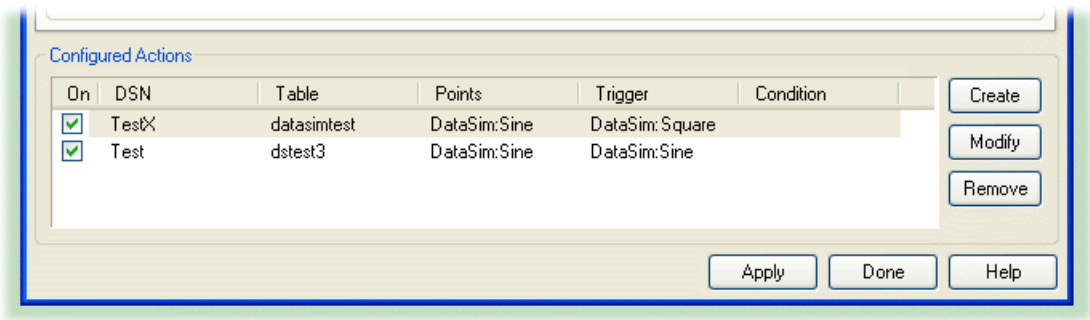
LogFunctions.g

```
/*
 * Functions that can be used by the Condition section of a Data Logging
 * configuration to filter whether data is actually written to the
 * database when a trigger occurs.
 */

// Change this to nil to silence the debugging statements that will be written
// to the script log.
LogFunctionDebug = t;

 /*
  * Track the previous value of a DataHub point and only allow the logging
  * action if the DataHub point value has changed since the last time we
  * logged it.
  *
  * Takes a single argument that is either a point symbol or a point name as
  * a string.
  *    e.g., "DataPid:PID1.Mv"
  *          #$DataPid:PID1.Mv
  *
  * Notice that a point symbol must be specified with the # operator before the
  * symbol to ensure that the symbol is passed, not the value of the point.
  */

function HasValueChanged(point_symbol)
{
    local ptsym = symbol(point_symbol);
    local oldvalue = getprop(ptsym, #last_log_value);
    local curvalue;
    local result = nil;

    if (!undefined_p(curvalue = eval(ptsym)))
```

```
    {
        if (LogFunctionDebug)
            princ("Checking value of ", ptsym, ": previous = ", oldvalue,
                  ", current = ", curvalue, "\n");
        if (oldvalue != curvalue)
            result = t;
        setprop(ptsym, #last_log_value, curvalue);
    }
    else
    {
        if (LogFunctionDebug)
            princ("Log condition: value of ", ptsym, " is undefined\n");
    }
    result;
}
```

7.9. Configured Actions

A *configured action* will cause a row of data to be written into the table specified in your DSN, based on a trigger and optional conditions. It is the end result of your configuration activities in this interface. The Configured Actions list shows the actions you have configured, and allows you to create, modify, or remove actions, as well as turn them on or off.



The list of configured actions shows the actions you have already configured. Selecting an existing action from the list automatically fills in the DSN, Table, Trigger, and Condition tabs with its information. Checking or unchecking the On box at the left lets you switch the action on or off.

The Create button creates an action for the information currently entered in the DSN, Table, Trigger, and Condition tabs. If you press the Create button while a configured action is selected, it creates a duplicate of that configured action and adds it to the list. This is a quick way to configure similar actions.

The Modify button overwrites the selected configured action with the information currently entered in the DSN, Table, Trigger, and Condition tabs.

The Remove button removes a configured action.

Once a configured action has been created or modified, the changes won't take effect until you click the Apply or Done button. Each write action to the database gets logged in the VINE Script Log. This allows you to check for error messages and ensure that your data is being written successfully. You can also verify the writes by querying the database itself.

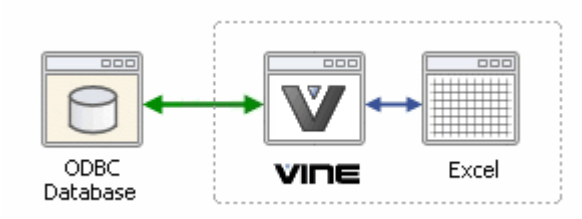
Notes

1. Text logging can be configured through VINE scripting.

Chapter 8. Query a Database

8.1. Introduction

Skkynet VINE can query any ODBC compliant database, and bring the data back as VINE points, or into a single point in XML format for use in WebView.

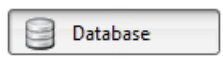


Skkynet VINE's ODBC Data Logging interface provides an easy way to connect to a database and submit a query. There are two options for writing the query to VINE: one VINE point per row, or the whole query results in a single VINE point. The fastest way to learn how to use the interface is by using the Quick Start.

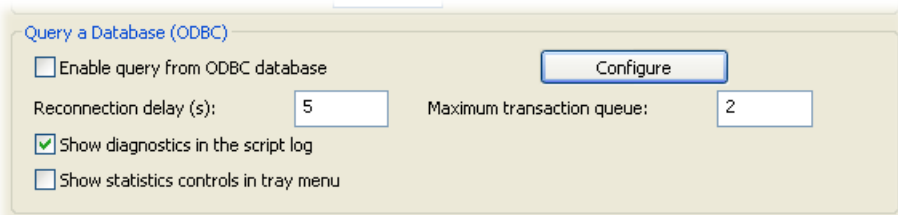
8.2. Quick Start

Here's how to configure Skkynet VINE to query a database of your choice.

Open the Query ODBC Database window

1. In the Skkynet VINE Properties window, select Database 

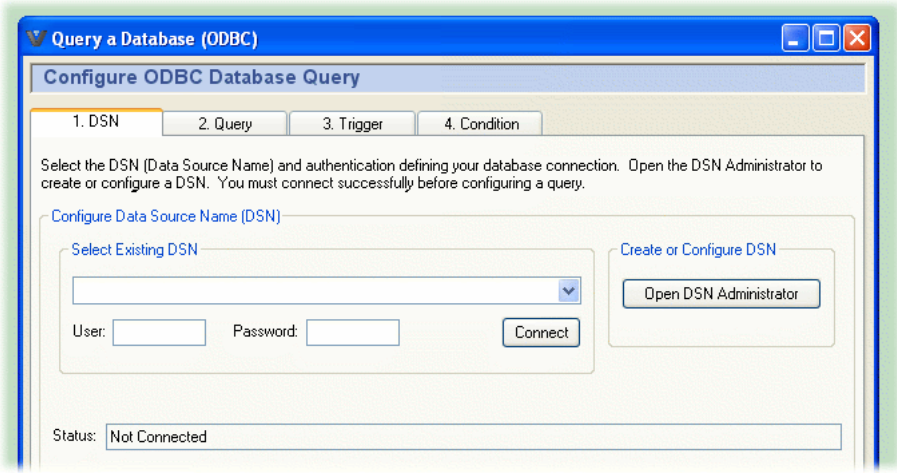
.
2. In the Query a Database (ODBC) section, click the Configure button.



This opens the Query a Database (ODBC) window, shown below.

Connect to the Database

1. Select the 1. DSN tab. A *DSN* is a Data Source Name. Windows uses this name to identify the database you want to connect to.


Query a Database (ODBC)
Configure ODBC Database Query
1. DSN
2. Query
3. Trigger
4. Condition
Select the DSN (Data Source Name) and authentication defining your database connection. Open the DSN Administrator to create or configure a DSN. You must connect successfully before configuring a query.
Configure Data Source Name (DSN)
Select Existing DSN
Create or Configure DSN
Open DSN Administrator
User:
Password:
Connect
Status:
Not Connected

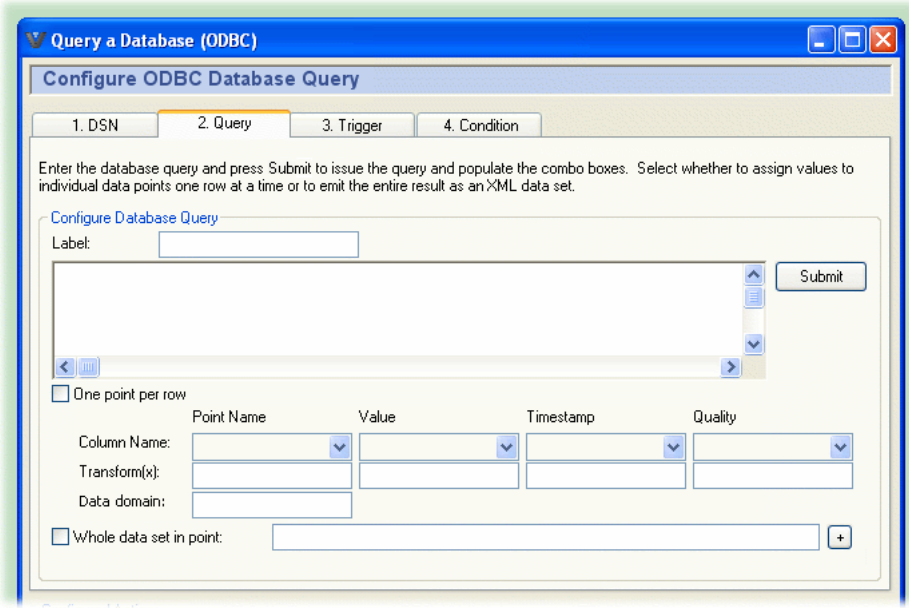
2. From the drop-down box, select a DSN. If you do not have any DSNs, or you wish to create a new DSN, you can do this by opening the DSN Administrator. Please refer to Setting up a DSN for more details.
3. Enter the appropriate user name and password (if required), and click the Connect button. A "`Connected to ...`" message should appear in the message box. If you get an error message in the box, consult your system administrator.

Configure a Query

For this and other queries in this chapter, we will use a simple example database named `test` with a table named `querytest` that has 4 columns and 3 rows.

In our example, we will be writing the contents of the VariableName and VariableValue columns to VINE

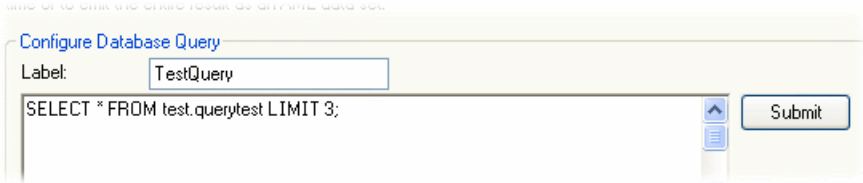
1. Select the 2. Query tab.


Query a Database (ODBC)
Configure ODBC Database Query
1. DSN
2. Query
3. Trigger
4. Condition
Enter the database query and press Submit to issue the query and populate the combo boxes. Select whether to assign values to individual data points one row at a time or to emit the entire result as an XML data set.
Configure Database Query
Label:
Submit
One point per row
Point Name
Value
Timestamp
Quality
Column Name:
Transform(x):
Data domain:
Whole data set in point:

2. In the Label: field, enter a name for this query, such as **`TestQuery`**. This can be any string.
3. Enter a valid SQL query. As an example, for our small database, we can use a simple SQL query:

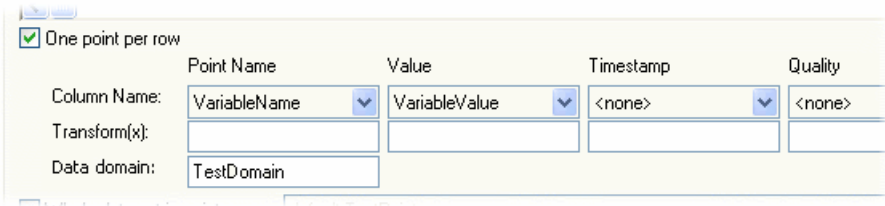
```
SELECT * FROM database.table LIMIT 3;
```

If you use a `SELECT * FROM` query like this on a large database with a frequent timer, we recommend limiting the number of rows returned, to prevent hanging the database.

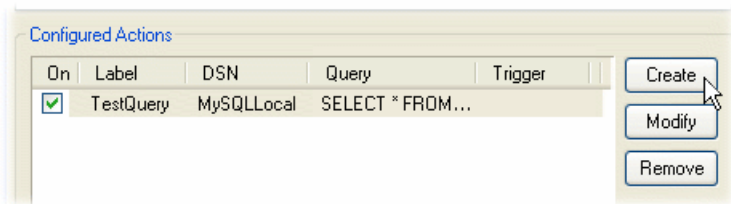


4. Press the Submit button to submit your query to the database for a check. If the query is not valid, a message will pop up informing you about any errors. You can also open the Script Log to see more information about your connection to the database, and the results of your query.
5. There are two options for how a query is written to VINE. Here we will introduce the One point per row option. For information about the Whole data set in a point option, please refer to Section 8.4, *Configuring a Database Query*.

   Select One point per row.



6. In the Column Name / Point Name dropdown list, choose the column name in your database that contains the point names that you will be using. For instance, in our demo database that column is named VariableName.
7. In the Column Name / Value dropdown list, choose the column name in your database that contains the values for the point names that you will be using. For instance, in our demo database that column is named VariableValue.
8. In the Replace domain field, enter the name of a new or existing domain. We've put in a name for a new domain, TestDomain.
9. Click the Create button.



   A new configured action should appear in the list. For more information about configured actions, please refer to Section 8.7, *Configured Actions*

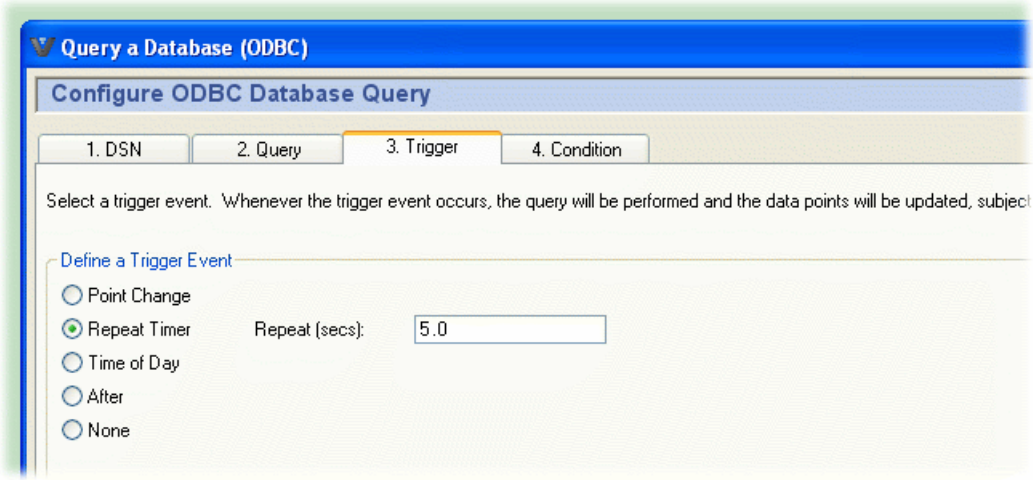
   Next, to get VINE to make the query, you need to assign a trigger.

Assign a Trigger

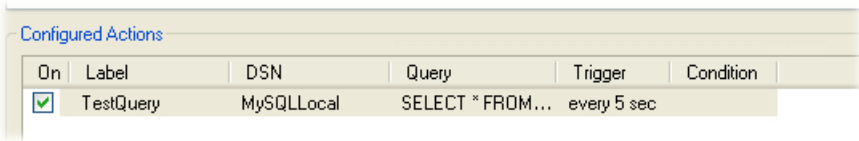
For this example, we will set a repeat timer to requery the database every 5 seconds.

1. Select the 3. Trigger tab.

2. Select Repeat Timer and enter **5**.



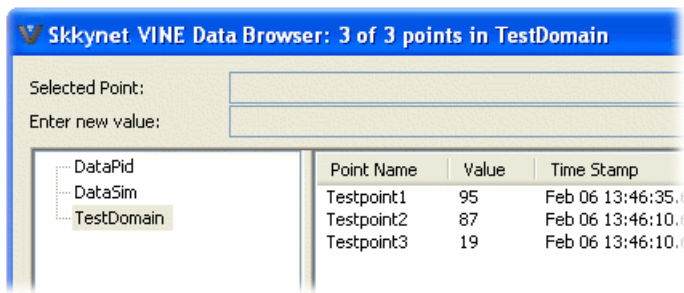
3. In the Configured Actions box, make sure that the configured action you just created is highlighted. If not, click on it to highlight it. Then click the Modify button.



   Your configured action should now display every 5 sec in the Trigger column.

4. Click the Apply button to activate the configured action.

5. Open the Data Browser window. After 5 seconds, the points should appear in the Data Browser:



   To test the updates, you can change a value associated with a point name in the database, and the change should appear in the Data Browser within 5 seconds.

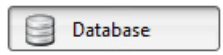
   Should you not see the data points, or changes made to the values, you can open the Script Log to check for error messages and ensure that your query is being sent every 5 seconds.

If all is working as described, you have configured an action that queries a database for and writes the results into VINE. The remaining sections in this chapter explain the interface in more detail, and introduce the option of setting specific conditions for queries, if desired.

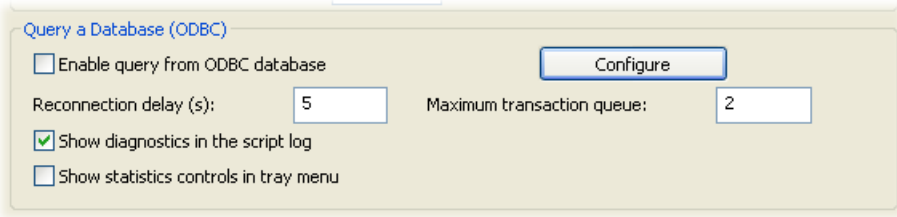
8.3. Setting up the DSN (Data Source Name)

A *DSN* is a Data Source Name. Windows uses this name to identify the database you want to connect to. This tab lets you select an existing DSN, or create a new one if necessary.

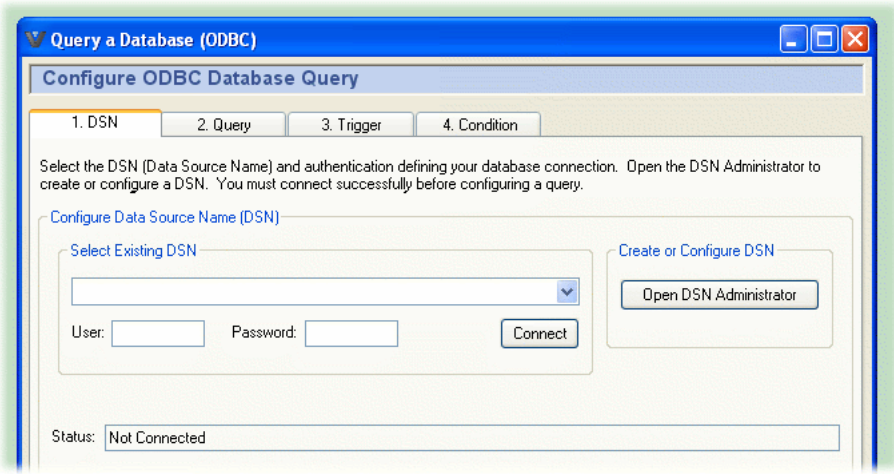
Open the Query a Database (ODBC) window

1. In the Skkynet VINE Properties window, select Database 

.
2. In the Query a Database (ODBC) section click the Configure button.



3. Configure your DSN as explained below.


Query a Database (ODBC)
Configure ODBC Database Query
1. DSN
2. Query
3. Trigger
4. Condition
Select the DSN (Data Source Name) and authentication defining your database connection. Open the DSN Administrator to create or configure a DSN. You must connect successfully before configuring a query.
Configure Data Source Name (DSN)
Select Existing DSN
Create or Configure DSN
Open DSN Administrator
User:
Password:
Connect
Status:
Not Connected

Selecting a DSN

1. To select a DSN, choose one from the drop-down box, and then enter the appropriate user name and password.

   If your DSN is using Windows Authentication, leave the User and Password fields blank.

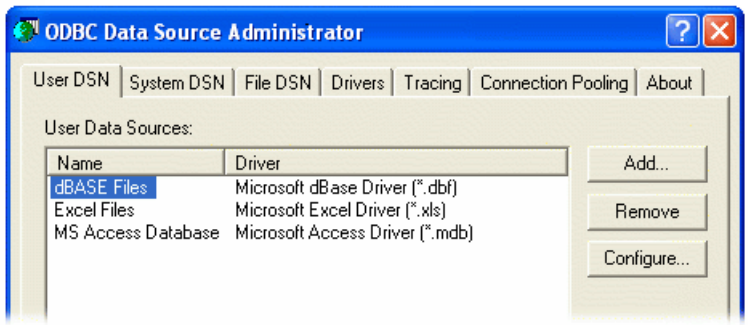
2. Click the Connect button. A "`Connected to ...`" message should appear in the message box. If you get an error message in the box, consult your system administrator.

Creating or Configuring a DSN

Windows has different configuration programs for 32-bit and 64-bit DSNs. Skkynet VINE uses 32-bit DSNs. To configure a 32-bit DSN in a 64-bit system, you need to run:

```
c:\windows\syswow64\odbcad32.exe
```

1. To create or configure a DSN, click the Open DSN Administrator button. This opens the ODBC Data Source Administrator window.



2. Select the User DSN or System DSN tab, depending on how you plan to access your database.

   A user DSN is only available to the current user account, while a system DSN is available to any user account on the computer.

3. Now you can add a new database or configure an existing one.

   Add a new database

   1. Click the Add button. The Create New Data Source window will open, displaying a list of data source drivers.
   2. Select the data source driver that corresponds to your ODBC database. A data source setup window will open. Each data source setup window is different, but you should be able to find the appropriate entry fields easily enough.
   3. Enter the data source name and select the database.
   4. Enter any other required or optional information such as login name, password, etc. What entries need to be made and where they are entered depends on the particular data source setup window you are using.
   5. Click OK to return to the ODBC Data Source Administrator window. You should be able to see the new database and driver listed. If you need to make any changes, you can configure an exiting database, as explained below.

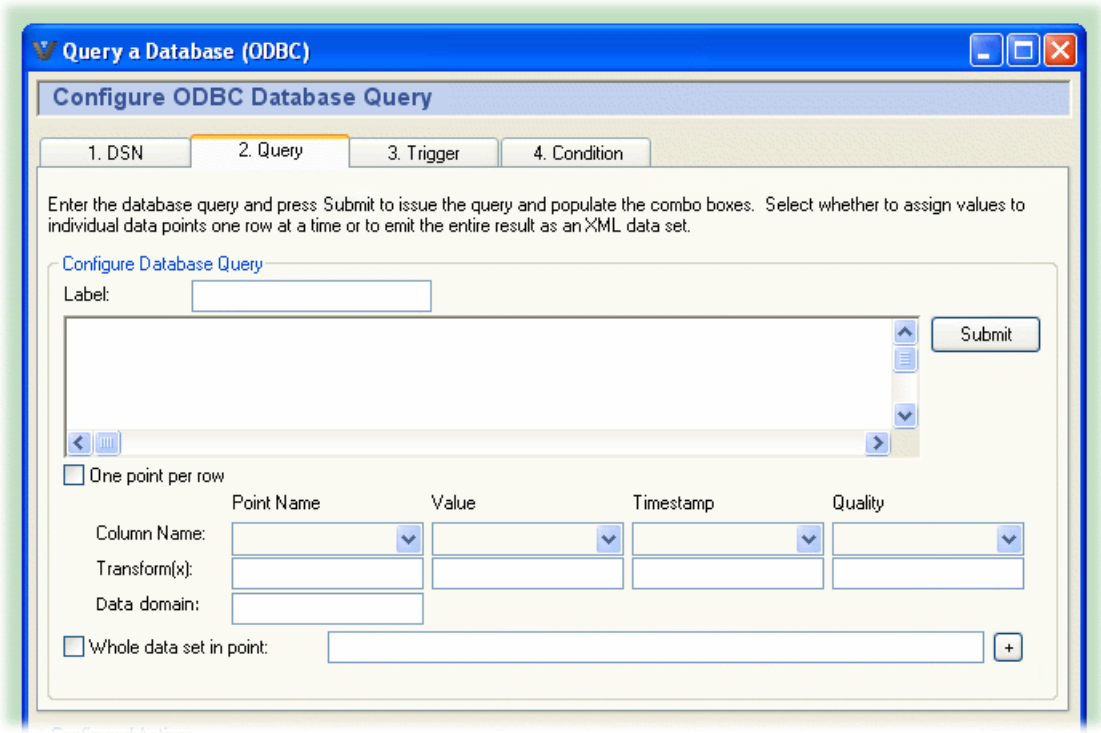
   Configure an existing database

   1. Select a data source name and click the Configure... button. This takes you to the data source setup window (explained above) where you can make changes to the configuration.
   2. Make your changes and click OK to return to the ODBC Data Source Administrator window. Any time you need to make a change, you can go to this window.

Once you have created a DSN, you can select it as explained above.

8.4. Configuring a Database Query

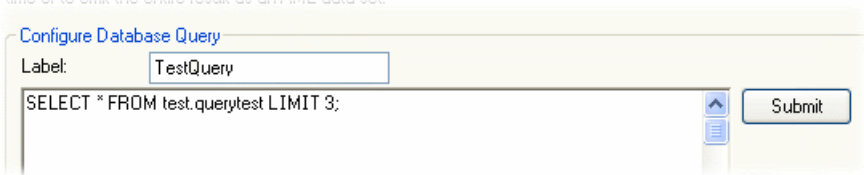
After you have set up a DSN you can create a database query.


Query a Database (ODBC)
Configure ODBC Database Query
1. DSN
2. Query
3. Trigger
4. Condition
Enter the database query and press Submit to issue the query and populate the combo boxes. Select whether to assign values to individual data points one row at a time or to emit the entire result as an XML data set.
Configure Database Query
Label:
Submit
One point per row
Point Name
Value
Timestamp
Quality
Column Name:
Transform(x):
Data domain:
Whole data set in point:

Preparing the Query

For this and other queries in this chapter, we will use a simple example database named `test` with a table named `querytest` that has 4 columns and 3 rows.

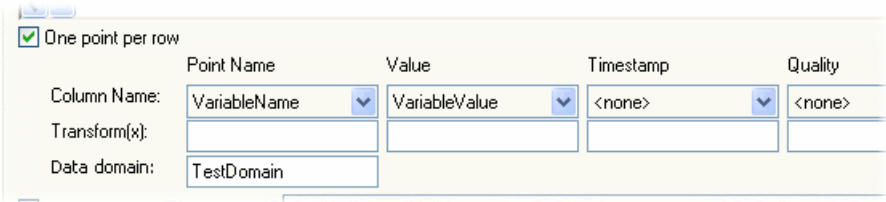
- In the Label: field, enter a name for this query, such as **`TestQuery`**. This can be any string.
- Enter a valid SQL query.



- The Submit submits your query to the database for a check. If the query is not valid, a message will pop up informing you about any errors. You can also open the Script Log to see more information about your connection to the database, and the results of your query.

Write One Point Per Row to VINE

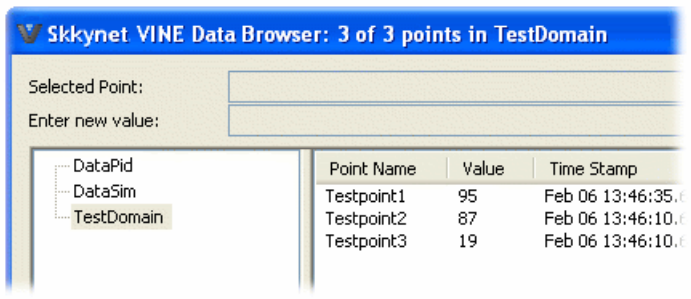
- The One point per row option lets you assign one point in VINE for each uniquely named item in one column of a database.



- The drop-down Column Name lists allow you to chose the columns that contain the point name and its value that will appear in VINE, as well as an associated time stamp and quality.
- The Transform(x) fields let you to perform transformations on the data in each row of the corresponding column. These transformations are written in Gamma, the VINE Scripting language.

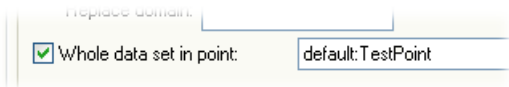
  For example, an entry of `string("MyTag", x)` under the Point Name would change points labelled `Pump1` and `Pump2` to `MyTagPump1` and `MyTagPump2`. Or, an entry of `10 * x` under the Value would multiply each value by `10`.

- The output of One point per row for our example database would appear in VINE Data Browser like this:



Put a Whole Data Set into One VINE Point

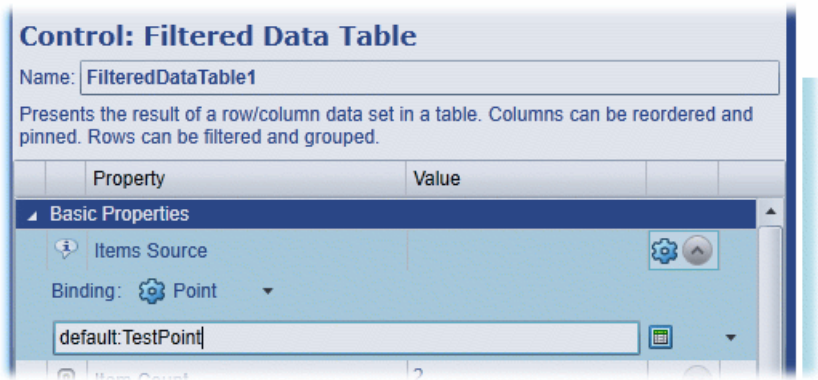
The Whole data set in point option lets you put the entire results of the query into one VINE point, as specified, formatted in XML.



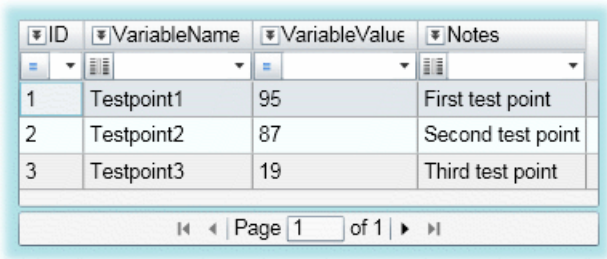
For example, our query of `SELECT * FROM test.querytest LIMIT 3;` on our example database would write the following XML-formatted data as the value of a single point:

```
<TableData>
  <column name="ID" type="int" datatype="-6" />
  <column name="VariableName" type="string" datatype="-9" />
  <column name="VariableValue" type="double" datatype="8" />
  <column name="Notes" type="string" datatype="-9" />
  <row>
    <ID>1</ID>
    <VariableName>Testpoint1</VariableName>
    <VariableValue>95</VariableValue>
    <Notes>First test point</Notes>
  </row>
  <row>
    <ID>2</ID>
    <VariableName>Testpoint2</VariableName>
    <VariableValue>87</VariableValue>
    <Notes>Second test point</Notes>
  </row>
  <row>
    <ID>3</ID>
    <VariableName>Testpoint3</VariableName>
    <VariableValue>19</VariableValue>
    <Notes>Third test point</Notes>
  </row>
</TableData>
```

This data can be parsed by an XML parser, and is particularly useful for displaying in WebView, using the WebView Filtered Data Table control. In WebView, you would add a Filtered Data Table to a page, and configure this point as the Items Source:

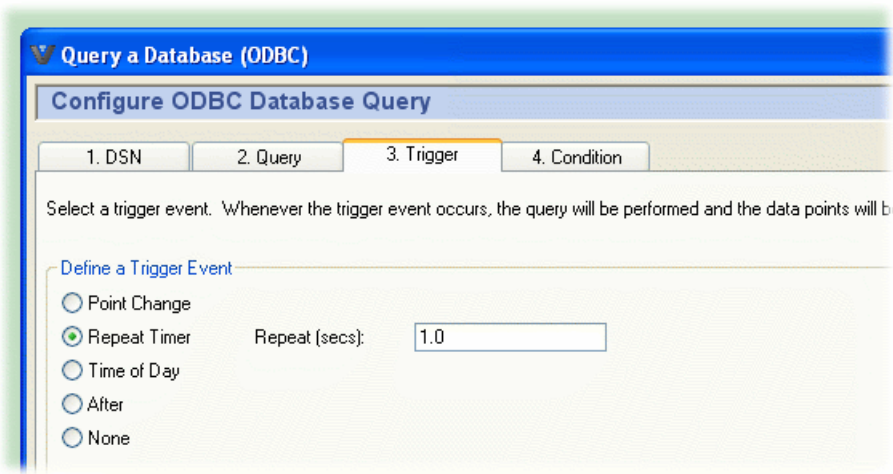

Control: Filtered Data Table
Name: FilteredDataTable1
Presents the result of a row/column data set in a table. Columns can be reordered and pinned. Rows can be filtered and grouped.
Property
Value
Basic Properties
Items Source
Binding: Point
default:TestPoint

The data would then appear in the control:



8.5. Assigning a Trigger

A trigger is an event that causes your query to be sent to the database. A trigger event can be either a point value change, a repeat timer, a time of day timer, or an "after" timer. You can assign a different trigger for each query, or an identical trigger to any number of queries. A query action can be configured to execute on every trigger event, or you can assign trigger conditions that are evaluated whenever a trigger occurs, to determine if the action should be executed.



The four kinds of triggers are:

- **Point Change** fires whenever a specified trigger point changes.

  1. Type the name of the point into the Point Name box, or select the point using the data tree on the right, then click the + button.
  2. (Optional) Enter a value deadband if you want to filter out extraneous data. The number you enter will specify a high and low (plus or minus) range. Any value change falling within that range will not cause the trigger to fire. A positive or negative change greater than this value will activate the trigger and cause the row to be written.

  To create a trigger that gets reset automatically, please refer to An Auto-Resetting Trigger in Section 8.6, *Setting Trigger Conditions*.

- **Repeat Timer** fires cyclically, each time the specified number of seconds elapses.
- **Time of Day** fires at the time you specify. You can enter:
  - A number, indicating a specific value. For example, a `0` in the seconds field would cause the event only on the 0th second of the minute. A `30` would indicate only on the 30th second of the minute.
  - A list of numbers, separated by commas. For example, entering `0,15,30,45` in the minutes field would indicate that the event should fire on the hour and at 15, 30 and 45 minutes past the hour.
  - A range of numbers, separated by a dash. For example, entering `8-18` in the hours field would indicate that the event should fire every hour from 8 a.m. to 6 p.m.. Ranges can be mingled with lists, as in `0,4,8-16,20`.
  - An asterisk (`*`) indicates that the event should fire for every possible value of the field. For example, a `*` in the seconds field would cause the event to fire every second. A `*` in the hours field would cause the event to fire every hour.

  To regularly log a record on specific days of the week, please refer to Section 8.6, *Setting Trigger Conditions*.

  The ranges of the fields are:

Year:	`1970-*`	Hour:	`0-23`
Month:	`1-12`	Minute:	`0-59`
Day:	`1-31`	Second:	`0-59`

  The year and month are entered differently here than for the Gamma `localtime` function, as explained in Time Conditions.

  Examples:

  - These entries:

    would trigger a query at 8:45 every day, every month, and every year.

  - These entries:

would trigger a query every hour on the 15th day of each month, every year.

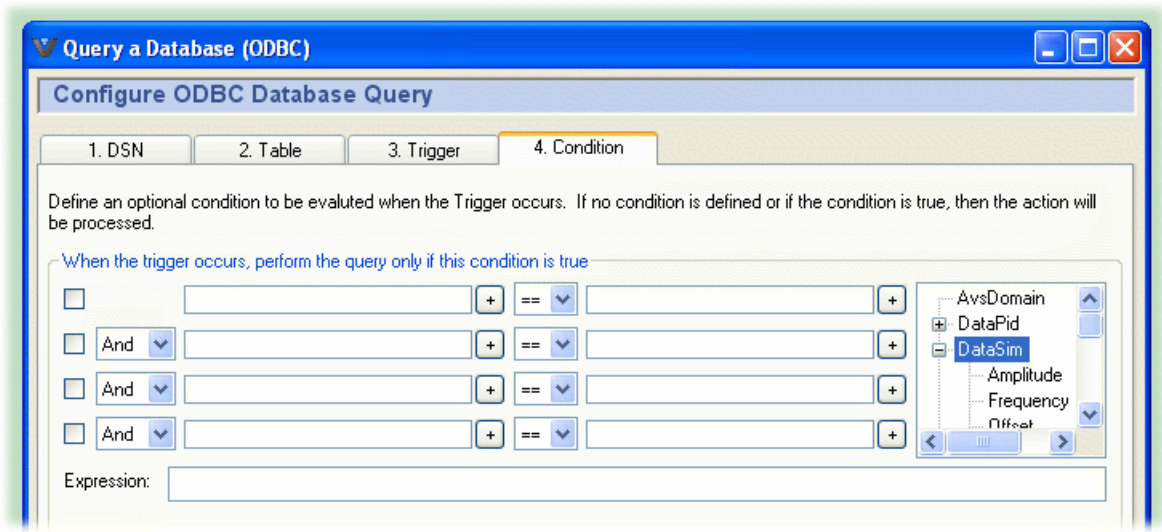
- These entries:

would trigger a query every second for 5 minutes, every two hours between 8 a.m. and 6 p.m.

- **After** fires once, when the specified number of seconds elapses.
- **None** configures no trigger.

8.6. Setting Trigger Conditions

Each query can have up to four conditions that determine whether it gets sent to the database when the trigger fires.



Fill in the conditions according to the guidelines below. Check the box next to the condition to apply it. As you make entries, the corresponding *Gamma* code will appear in the display. Gamma is VINE's built-in scripting language. The code that appears in the Expression box is the actual code that gets run by Gamma. The order of precedence for "`And`" and "`Or`" operators (&& and `||`) is first `And`, then `Or`.

Point Value Conditions

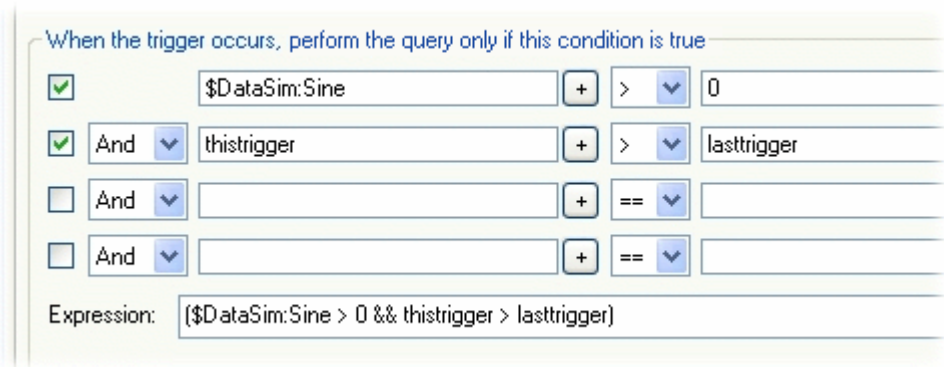
Point names can be entered on either or both sides of the comparison. They can be picked from the data tree list, or typed in. Each point name needs to have a dollar sign ($) in front of it to indicate to Gamma that this is a VINE point. You can put numerical values into either side of the comparison.

When you enter a point name in a condition field, the current value of the point will be used in the evaluation. For example, you could define a condition that states that whenever the trigger event occurs, the action will only be executed if another point value is within a certain range.

There are some automatic variables available for working with point values:

- `lasttrigger` - the value of the trigger point the last time this trigger was fired (even if the condition failed).
- `thistrigger` - the value of the trigger point now (even if the condition failed).
- `lastevent` - the value of the trigger the last time the event was actually executed (the condition succeeded).
- `this` - the trigger point itself, not the value of the point.

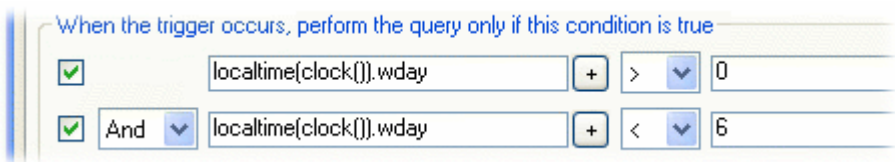
You can use these variables in any condition that is triggered by a data value change. For example, you could create some conditions like this:



Time Conditions

This provides an additional way to restrict the time, day, month, etc. when a query gets sent. In addition to the options on the triggers, here you have day-of-week condition statements which can give you more flexibility for evemts based on specific days of the week. These will work with any type of trigger event.

You can use the Gamma functions `clock` and `localtime` to specify particular days of the week. For example, these entries:



would create this Gamma code::

```
(localtime(clock()).wday > 0 && localtime(clock()).wday < 6)
```

which would cause a query to be sent only Monday through Friday. The function `localtime` returns a class whose members contain information about the date, as follows:

`.sec`	The number of seconds after the minute (`0` - `59`).
`.min`	The number of minutes after the hour (`0` - `59`).
`.hour`	The number of hours past midnight (`0` - `23`).
`.mday`	The day of the month (`1` - `31`).
`.mon`	The number of months since January (`0` - `11`)
`.year`	The number of years since 1900.
`.wday`	The number of days since Sunday (`0` - `6`).
`.yday`	The number of days since January 1 (`0` - `365`)
`.isdst`	`1` if daylight saving time is in effect, `0` if not, and a negative number if the information is not available.

The year and month are entered differently here than for Time of Day trigger conditions, as explained in Section 8.5, *Assigning a Trigger*.

There are two automatic variables available for working with time values:

- `lasteventtime` - the system clock time (UTC floating point) that the last event was executed.
- `curtime` - the system clock time (UTC floating point) when the current trigger occurred.

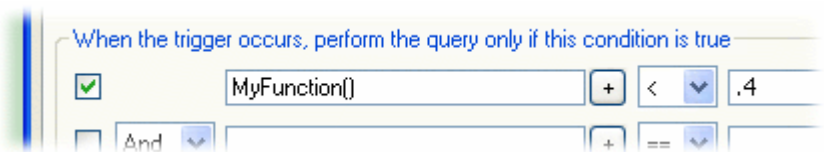
Custom Conditions

If the conditions you need to meet are beyond the scope of this interface, you can use a Gamma function to express virtually any condition you need. Then you can insert the function into one of the condition boxes, and set a condition based on the return value of the function.

To do this is you can create a VINE script (`.g` file) that contains only the functions you will be using for conditions, without any classes or methods. For example, here is the complete contents of such a file, named `MyConditions.g`:

```
function MyFunction ()
{
    myvalue = $DataSim:Sine;
    princ("Value when the trigger fired: ", myvalue, "\n");
    myvalue;
}
```

This function prints the value of the `DataSim:Sine` point, and returns its value. We can use this function as a condition by calling it from one of the condition boxes in the interface, like this:



When the trigger fires, `MyFunction` is called, and the return value gets checked to see if it is less than `.4`. If so, the data gets logged.

Example: An Auto-Resetting Trigger

This script can turn any VINE point into a trigger that automatically resets. To use it, you first need to load and run the `TriggerFunctions.g` script (shown below and included in the installation archive). Then, if you put this formula:

```
HighWithReset($TriggerPoint)!= nil
```

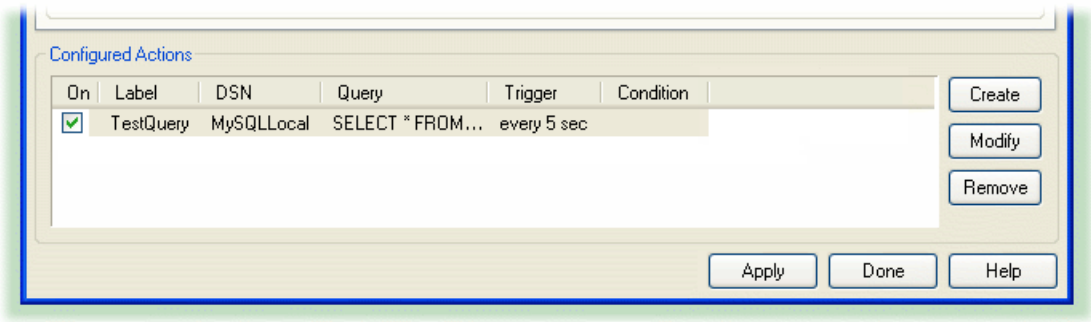
into the condition boxes, whenever the *TriggerPoint* changes to a non-zero number in VINE, your trigger will fire. The script waits for a millisecond, then resets the *TriggerPoint* back to zero. The second function works similarly, but triggers on a change to zero, instead of a change to a non-zero number.

TriggerFunctions.g

```
/*
 * This file contains handy functions to perform more complex condition
 * handling in the Condition tab of the data logging and email interfaces.
 */

/*
 * Test a trigger point for a non-zero value.  If the point is non-zero,
 * create a delayed event to reset the point to zero, and return true,
 * indicating that the condition has succeeded and the action should proceed.
 * If the value is 0, then simply return nil indicating that the action
 * should not proceed.  We need to test for zero because when we reset the
 * trigger point to zero a second data change event will occur.
 *
 * The argument is unevaluated, so the condition should look like this:
 *    HighWithReset($default:TriggerPoint) != nil
 */

function HighWithReset(!triggerpoint)
{
    local   value;
    if (!undefined_p(value = eval(triggerpoint)) && value != 0)
    {
        after(0.001, 'setq(@triggerpoint, 0));
        t;
    }
    else
    {
        nil;
    }
}

/*
 * This is the inverse of HighWithReset (see above).  If the trigger point
 * is zero, perform the action and set the trigger point to 1.  If the
 * trigger point is non-zero do nothing and return nil.
 */
function LowWithSet(!triggerpoint)
{
    local   value;
    if (!undefined_p(value = eval(triggerpoint)) && value == 0)
    {
        after(0.001, 'setq(@triggerpoint, 1));
        t;
    }
    else
    {
        nil;
    }
}
```

8.7. Configured Actions

A *configured action* will cause your query to be sent to the database, according to the trigger and any conditions you have specified. It is the end result of your configuration activities in this interface. The Configured Actions list shows the actions you have configured, and allows you to create, modify, or remove actions, as well as turn them on or off.



The list of configured actions shows the actions you have already configured. Selecting an existing action from the list automatically fills in the DSN, Query, Trigger, and Condition tabs with its information. Checking or unchecking the On box at the left lets you switch the action on or off.

The Create button creates an action for the information currently entered in the DSN, Query, Trigger, and Condition tabs. If you press the Create button while a configured action is selected, it creates a duplicate of that configured action and adds it to the list. This is a quick way to configure similar actions.

The Modify button overwrites the selected configured action with the information currently entered in the DSN, Query, Trigger, and Condition tabs.

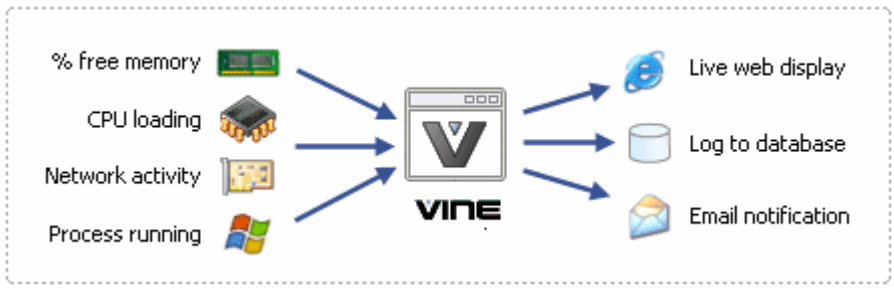
The Remove button removes a configured action.

Once a configured action has been created or modified, the changes won't take effect until you click the Apply or Done button. Each query to the database gets logged in the VINE Script Log. This allows you to check for error messages and ensure that your query was sent successfully. You can verify the results of the query in the Data Browser.

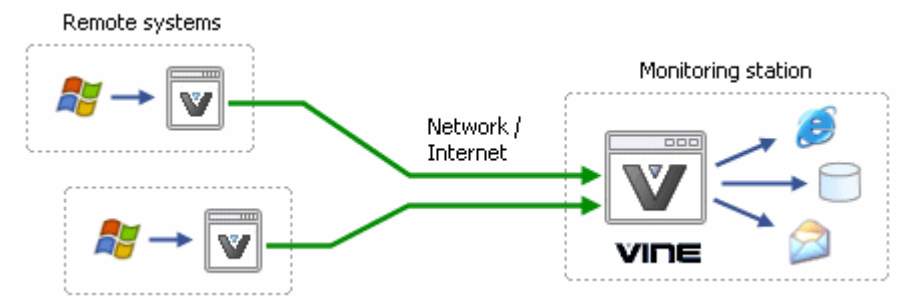
Chapter 9. System Monitor

9.1. Introduction

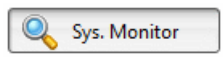
The Skkynet VINE System Monitor provides a way to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. in Skkynet VINE.



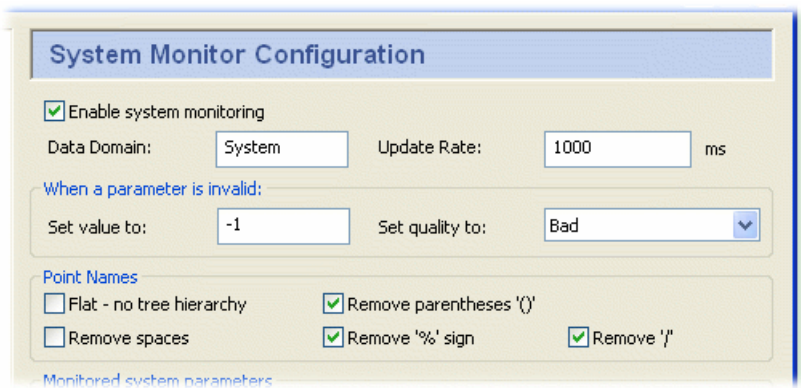
For example, by monitoring process ID you could determine whether a particular process is running or not. Any information accessed here becomes part of VINE's data set, and can thus be tunneled across the network, used in scripts or as email triggers, viewed in a spreadsheet, or stored in a database.



9.2. Configuring the System Monitor

1. With Skkynet VINE running, right click on the VINE system-tray icon and choose Properties.
2. In the Properties window, select System Monitor 

.



To enable system monitoring, check the Enable system monitoring box and edit the configuration options as desired:

Data Domain:

The name of any VINE data domain. The values retrieved from the system will be shown as points in this data domain.

Update Rate:

The frequency that the system is polled and all selected points are updated. The minimum polling time is 100 ms., so the value entered here cannot be less than `100`.

A high update rate (a low number here) for many data points could use a great deal of CPU.

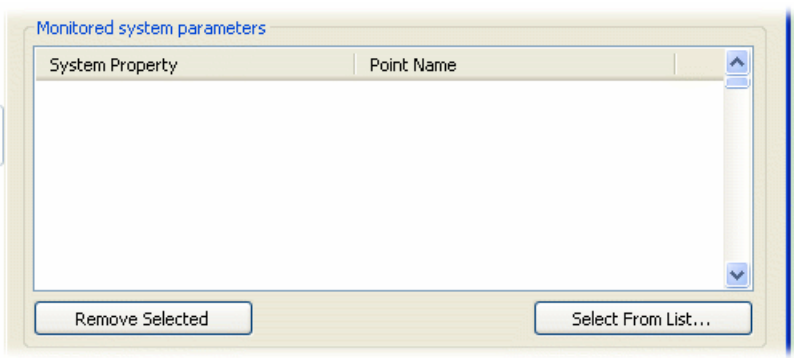
When a parameter is invalid:

A parameter will be invalid if the object being monitored is not available. For example, if a process is not running then the parameters for that process will all be invalid. This is a useful way to monitor a system process or other object. For example, you could use a script or other client to watch a process ID, and when the process ID becomes `-1` you could generate an alarm indicating that the process is no longer running.

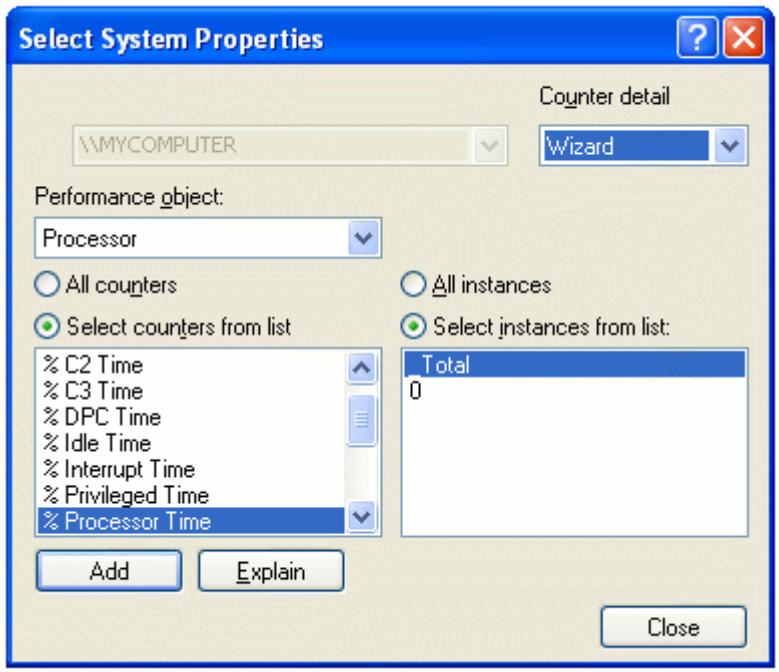
Point Names:

The System Monitor automatically creates Skkynet VINE point names based on the names of the system properties. Some client programs cannot work with point names containing special characters. This section allows you to specify which characters will be removed from the property name when constructing the point name.

Now you are ready to create the list of system parameters that you want to monitor.



3. Click the Select From List button. This will open the Select System Properties dialog:



Depending on your system, this dialog may take a few seconds to appear. If it does not come up, the Event Log will contain a message. Otherwise, just be patient, it will open eventually.

In the Select System Properties dialog you can specify which items to add to your list of monitored system properties, according to these criteria:

- **Performance object:** A list of all available objects, such as CPU, Memory, Process, Print Queue, TCP, etc.
- **Counters:** All of the available data categories related to the selected performance object. You can choose all counters, or select specific counters from the list. The Explain button opens a window with an explanation of the selected counter.
- **Instances:** All of the instances of the chosen performance object. For example, if you chose Process for your performance, this list will show all of the processes running on your system. You can choose all processes or select specific processes from the list.

  A number in this list normally indicates a selection from multiple objects of a given type, and `_Total` means the total across all of the objects. For example, if you are looking at `Processor` in a multi-processor machine, you will see a number (`0`, `1`, etc.) for each processor and a `_Total` for the cumulative statistic over all processors.

4. Select a performance object, and counters and instances as applicable. For example, to see the process ID for DataSim, first ensure that DataSim is running, then select:
   - **Performance object:** `Process`
   - **Select counters from list:** `ID Process`
   - **Select instances from list:** `DataSim`
5. Click the Add button to add the selected items to the Monitor system parameter list in the VINE Properties window.

6. Click the Apply or OK button in the Properties window when you are finished making your choices and filling the list, to apply your changes. You should be able to view the results in the Data Browser.

If you change your mind on what points to monitor, you can change the list at any time. Any points you remove from the list will continue to exist in VINE until it is shut down and restarted. Please refer to Section 13.1, *Data Points* for more information on creating and deleting points.

9.3. Monitoring Systems Across a Network

You can monitor a system across by using VINE *mirroring*. Mirroring is how two or more instances of Skkynet VINE link over a network or the Internet to maintain identical data sets.

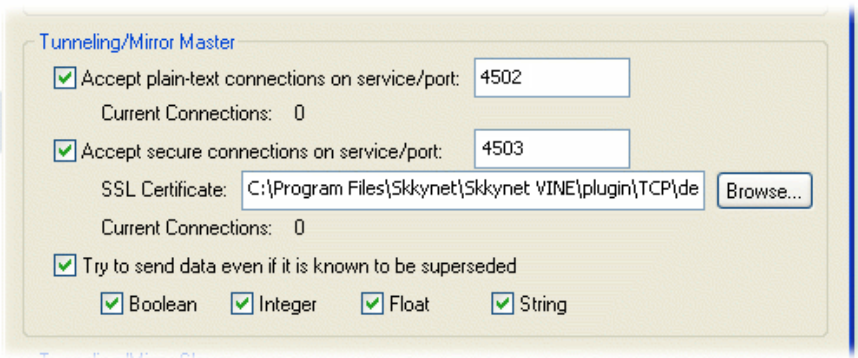
Mirroring is the same as *tunnelling*, as described in Chapter 4, *Tunnel/Mirror for Networking*. Mirroring can be also used to connect to Cascade DataHub running in Linux. Please refer to the Mirroring Data section of the Cascade DataHub for Linux and QNX manual for details.

For every mirroring connection, you must assign one VINE to be the master, and the other to be the slave. This determines which side initiates communication. Once communication is established, the data is identical. Generally it is recommended that VINE on the machine being monitored act as the master, while the machine that is collecting the monitoring data be the slave. In a hub-and-spoke arrangement, that VINE could be the slave to multiple masters, to collect all the data in a single VINE.

Configure VINE as a tunnel/mirror master

1. Right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.



3. In the Tunnelling Master section, you can configure plain-text or secure tunnelling. Ensure that at least one of these is checked. If you want to change any of the other defaults, please refer to Section 15.2, *Tunnel/Mirror* for more information.

To optimize throughput, un-check the Try to send data even if it is known to be superseded option. This will allow VINE to drop stale values for points which have already changed before the client has been notified of the original change. The latest value will always be transmitted.

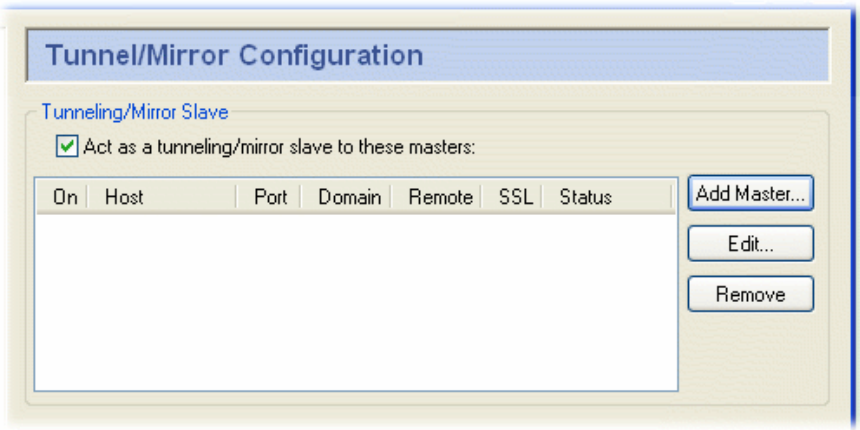
4. Click OK to close the Properties window.

You are now ready to configure the slave VINE.

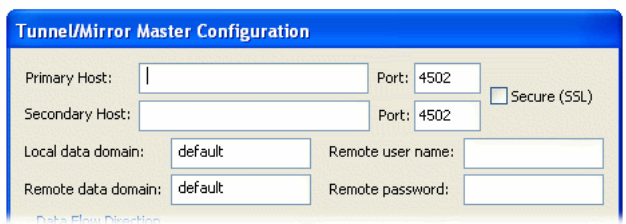
Configure Skkynet VINE as tunnel/mirror slave

1. Right click on the Skkynet VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.



3. Check the box Act as a tunnelling/mirror slave to these masters.
4. Click the Add Master... button to assign a master to this slave. The Tunnel/Mirror Master Configuration window will open:



5. Type in the following information:
   - **Primary Host:** the name or IP address of the computer running the tunnelling master VINE.
   - **Port:** the port number or service name for this host. You should use default port number (`4502`) unless you have changed the entry in the master VINE.
   - **Secondary Host:** gives you the option to have an alternate host and service/port number. On startup or after a network break, VINE will search first for the primary host, then for the secondary host, alternating between primary and secondary until a connection is made. If no secondary host is specified, the connection will be attempted on the primary host only.

     This feature is not recommended for redundancy because it only checks for a TCP disconnect. The VINE Redundancy feature, on the other hand, provides full-time TCP connections to both data sources, for instantaneous switchover when one source fails for any reason. There is no need to start up the OPC server and wait for it to configure its data set. You can also specify a preferred source, and automatically switch back to that data source whenever it becomes available. By contrast, the primary and secondary host in the tunnel can act as a primitive form of redundancy, but will only switch on a connection failure at the TCP level, which is only one sort of failure that a real redundancy pair must consider.

   - **Local data domain:** The data domain in which you plan to receive data.

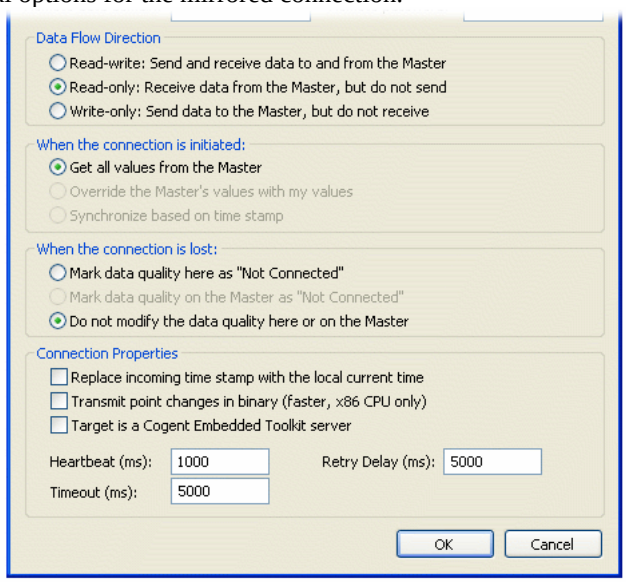
- **Remote data domain:** the master VINE data domain from which you plan to receive data. Point names will be mapped from the remote data domain (on the master VINE) into the local data domain (on this VINE), and vice versa.

  Unless you have a good reason for making these different, we recommend using the same data domain name on both VINEs for the sake of simplicity.

There is a VINE running on Cogent's server that you can connect to for testing. Here are the parameters you will need to enter for it:

- **Primary Host:** `developers.cogentrts.com`
- Port: `4502`
- **Local data domain:** `test`
- **Remote data domain:** `test`

6. You now have several options for the mirrored connection.


Data Flow Direction
Read-write: Send and receive data to and from the Master
Read-only: Receive data from the Master, but do not send
Write-only: Send data to the Master, but do not receive
When the connection is initiated:
Get all values from the Master
Override the Master's values with my values
Synchronize based on time stamp
When the connection is lost:
Mark data quality here as "Not Connected"
Mark data quality on the Master as "Not Connected"
Do not modify the data quality here or on the Master
Connection Properties
Replace incoming time stamp with the local current time
Transmit point changes in binary (faster, x86 CPU only)
Target is a Cogent Embedded Toolkit server
Heartbeat (ms): 1000
Retry Delay (ms): 5000
Timeout (ms): 5000
OK
Cancel

   a. **Data Flow Direction:** lets you determine which way the data flows. The default is bi-directional data flow between slave and master, but you can effectively set up a read-only or write-only connection by choosing that respective option.

      To optimize throughput, check the Read-only: Receive data from the Master, but do not send option. Only do this if you actually want a read-only connection. If you do not require read-write access, a read-only tunnel will be faster.

   b. **When the connection is initiated:** determines how the values from the points are assigned when the slave first connects to the master. There three possibilities: the slave gets all values from the master, the slave sends all its values to the master, or the master and slave synchronize their data sets, point by point, according to the most recent value of each point (the default).

   c. **When the connection is lost:** determines where to display the data quality as "Not Connected"—on the master, on the slave, or neither.

      If you have configured When the connection is initiated as Synchronize based on time stamp (see above), then this option must be set to Do not modify the data quality here or on the Master to get correct data synchronization.

d. **Connection Properties** gives you these options:

- **Replace incoming timestamp...** lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.
- **Transmit point changes in binary** gives users of x86 CPUs a way to speed up the data transfer rate. Selecting this option can improve maximum throughput by up to 50%.

  For more information, please refer to Section 14.1, *Binary Mode Tunnel/Mirror (TCP) Connections*.

- **Target is a Cogent Embedded Toolkit server** allows this slave to connect to an Embedded Toolkit server rather than to another VINE.
- **Heartbeat** sends a heartbeat message to the master every number of milliseconds specified here, to verify that the connection is up.
- **Timeout** specifies the timeout period for the heartbeat. If the slave VINE doesn't receive a response from the master within this timeout, it drops the connection. You must set the timeout time at least twice the heartbeat time.

  To optimize this setting for slow networks, please refer to Section 14.2, *Tunnel/Mirror (TCP) Connections for Slow Networks*.

- **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

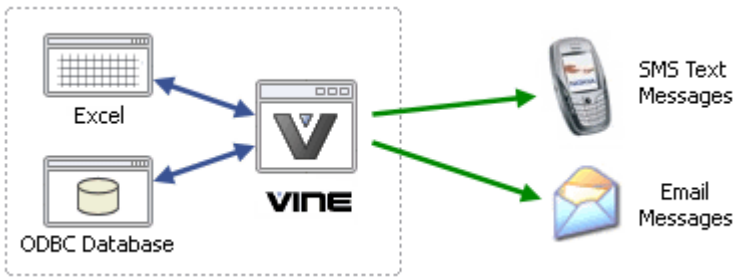
7. Click OK to close the Tunnel/Mirror Master window. The fields in the Tunnelling Slave table of the Properties Window should now be filled in.
8. Click the Apply button in the Properties Window. If the master VINE is running, this VINE should establish the tunnelling connection, and the Status should display `Connected`. You can view the data with the Data Browser, or view the connection with the Connection Viewer.

Open the Data Browser and select the data domain you requested to mirror. If the master Skkynet VINE has been correctly configured, you should now see all the master VINE data for that data domain.

Chapter 10. Email and SMS

10.1. Introduction

Skkynet VINE lets you send emails and SMS text messages, triggered by a VINE event such as a point value change, or by a timer. The emails and messages can be in plain text or HTML format, and they can contain current values for any data point in VINE



With this feature of the DataHub you can:

- Send SMS text messages to cell phones when an alarm event is triggered.
- Design end-of-day reports that are delivered to managers' email accounts each morning.
- Provide managers with regular email updates of production targets.
- Emails can contain data from OPC servers, ODBC databases and other sources.
- Eliminate errors associated with manually writing production reports.
- Have VINE collect vital report information, format it as an Excel spreadsheet and then email the file to key people for review.

10.1.1. How it works

The VINE has a built-in mailing program. VINE scripts tell the program what messages to send, to whom, and when to send them. A typical mailing script contains instructions to send an email in plain text, HTML format using ASP, or both. The example script, `MailTest.g` contains examples of both methods. You can run this script to test the mailer, and then use the examples that follow to send your own messages.

The sections in this chapter show you how to:

1. Configure the mailer.
2. Send a test message.
3. Send your own messages.
4. Create HTML messages.

10.1.2. A note about SMS text messages

Sending SMS text messages from the DataHub is done by simply sending a small plain text email to the appropriate SMS gateway email address. Most cell phone service providers offer email and text messaging options on new subscriptions.

For example, if you subscribed with Bell Mobility in Canada, and your new cell phone number was (416) 123 4567, then the email address for this phone would be:

```
4161234567@bellmobility.ca
```

and the SMS text message address would be slightly different:

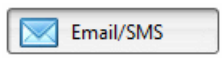
```
4161234567@txt.bellmobility.ca
```

Normally, SMS text messages are sent from one cell phone to another, which is all handled within the cell network itself. Using the SMS text message address, you can send a regular email and it will be converted by the SMS gateway (`txt.bellmobility.ca` in the example) into a text message and be delivered to your phone. There may be a short delay while the conversion from email to text message occurs, but the message usually arrives in less than a minute.

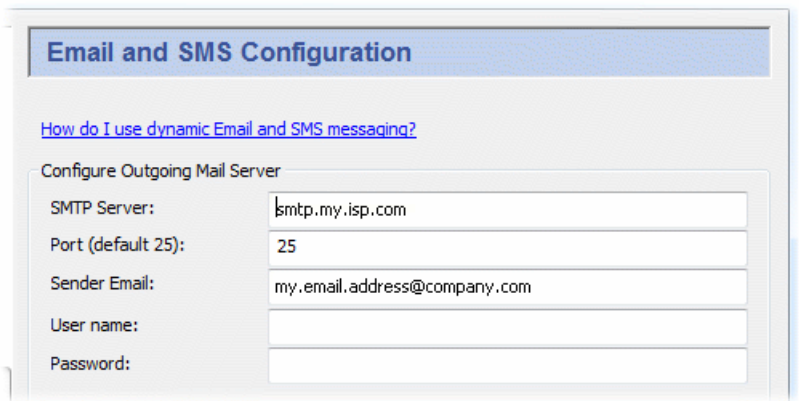
Check with your cell phone service provider for the SMS text message address for your cell phone number.

10.2. Configuring the Mail Server

Before you can send email from Skkynet VINE, you will need to configure the VINE mail server program, as follows:

1. With VINE running, right click on the VINE system-tray icon and choose Properties.
2. In the Properties window, select Email/SMS 

.



Enter the information that you want to use for sending the email. This can be the same as the SMTP server listed in your email client program.

SMTP Server:

The name of the SMTP server.

Port:

The SMTP port number (typically this is port 25).

Sender Email:

The email address of the sender. This will appear in the `From` field of the email. The address can be in either of these two forms:

- **`username@datadomain.com`** will be displayed as `username@datadomain.com` in the email reader (client).

- **`User Name <username@datadomain.com>`** will be displayed as `User Name` in the email reader (client).

User name:

The log-in name you use to access this SMTP account.

Password:

The applicable password.

3. In the Security section:

Security
Never attempt to connect securely via SSL
Always use SSL (fail if unavailable)
Automatically select most secure connection
Accept invalid or untrusted certificates

Choose one of the three SSL options, and specify whether you want to accept invalid or untrusted security certificates.

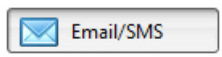
4. Click the Apply or OK button to submit your entries.

The VINE mailer is now ready to use. If you haven't already done so, we suggest sending a test message as explained in the next section.

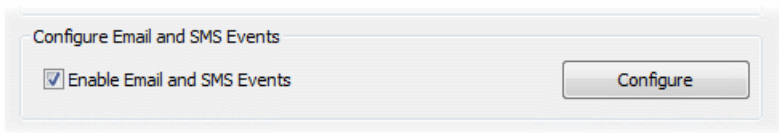
10.3. Sending a Test Message

Once you have configured the mail server you can configure and send a test email. Here's how:

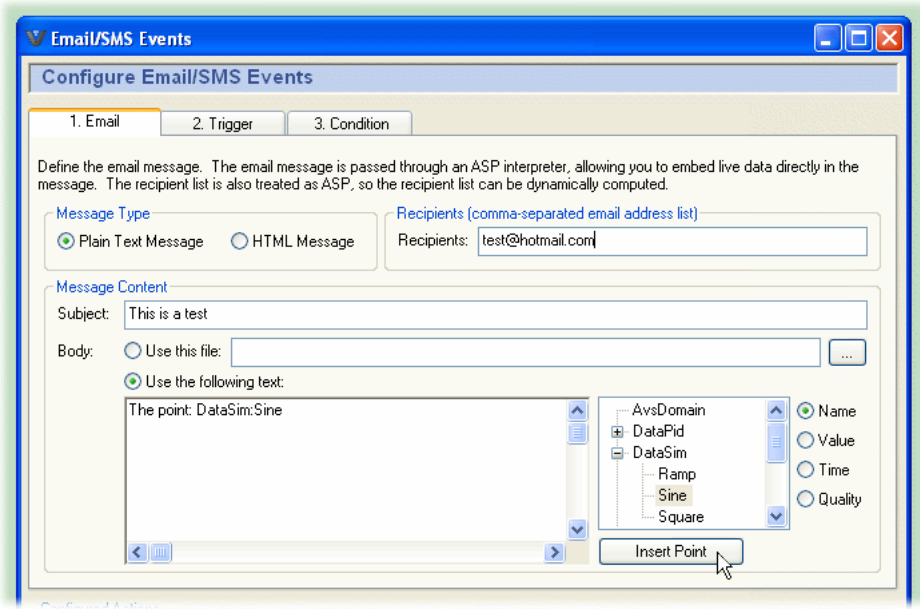
Open the Email/SMS Events window

1. In the Skkynet VINE Properties window, select Email/SMS 

.
2. In the Configure Email and SMS Events section press the Configure button.



This opens the Email/SMS Events window:



Define the Email Message

1. Select the 1. Email tab.
2. For the Message Type, choose Plain Text Message.
3. Enter a recipient email address in the Recipients box. You can enter several addresses, separated by commas.
4. Enter a subject in the Subject box.
5. For the Body, choose Use the following text:.
6. Start the DataSim program if it isn't already running, and ensure that it is connected to VINE.
7. In the point-picker list on the right, expand the `DataSim` data domain and select the point named `Sine`.
8. Click the Name button to the right of the point-picker list.
9. In the text entry field, type the following:

   ```
   The point
   ```

10. Click the Insert Point button. Your text display should now look like this:

    ```
    The point DataSim:Sine
    ```

11. Press **Enter** and continue typing:

    ```
    The point DataSim:Sine
    had a value of:
    ```

12. Click the Value button and then click the Insert Point button. Your text display should now look like this:

    ```
    The point DataSim:Sine
    had a value of: <%=$DataSim:Sine%>
    ```

13. Press **Enter** and continue typing:

    ```
    The point DataSim:Sine
    had a value of: <%=$DataSim:Sine%>
    at the time:
    ```

14. Click the Time button and then click the Insert Point button. Your text display should now look like this:

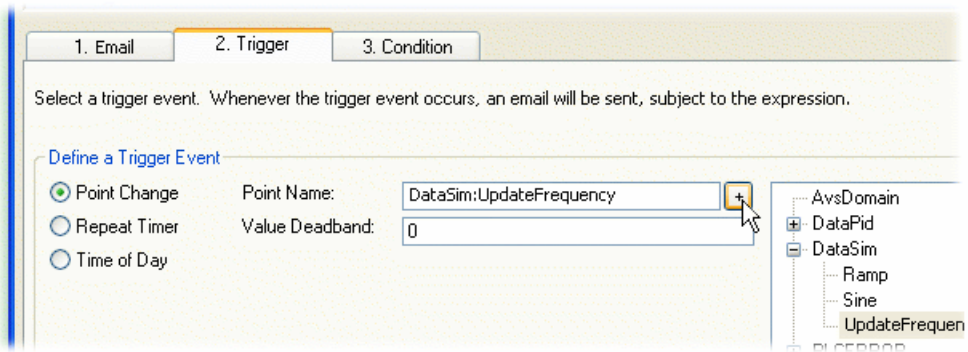
```
The point DataSim:Sine
had a value of: <%=$DataSim:Sine%>
at the time: <%=PointTimeString(#$DataSim:Sine%)>
```

The message is ready. Now you can assign a trigger and set a condition.

Assign a Trigger

For this example, we will trigger the action on the `DataSim:UpdateFrequency` point.

1. Select the 2. Trigger tab.
2. From the point selector, expand the `DataSim` data domain and select the point `UpdateFrequency`.
3. Click the + button to the right of the Point Name field. The point name `DataSim:UpdateFrequency` should fill in for you.

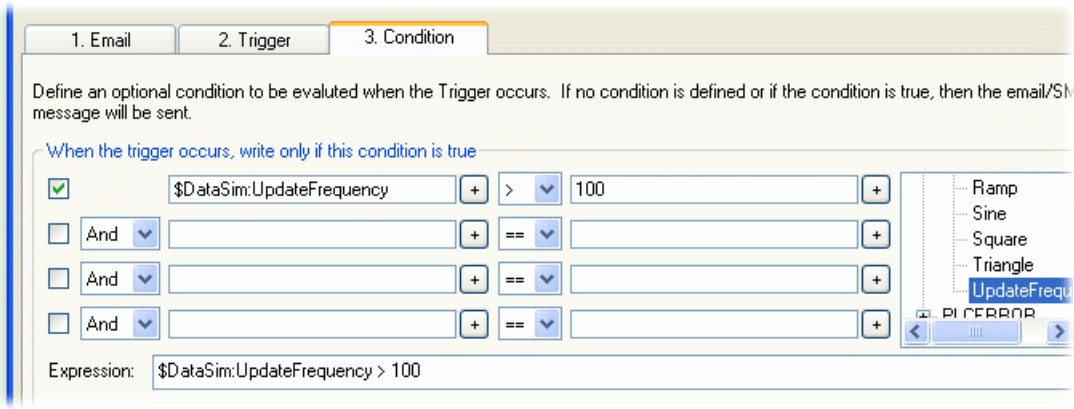


You can choose any point for the trigger, including the point that gets written, such as `DataSim:Sine` in our example. For more information about triggers, please refer to Section 10.5, *Assigning a Trigger*.

Set a Condition and Configure the Action

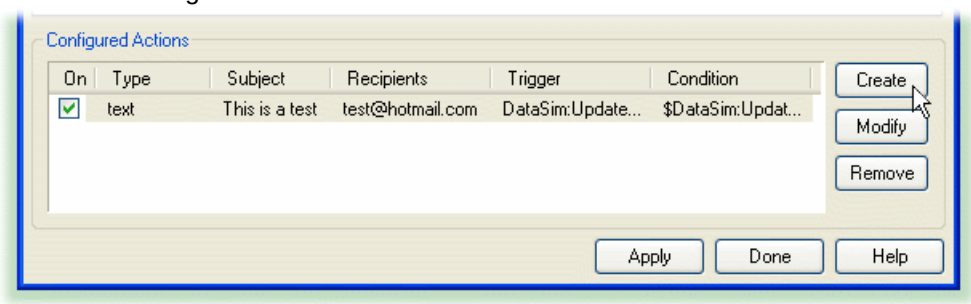
For this example, let's limit the trigger on the `DataSim:UpdateFrequency` point to changes only to values over `100`.

1. Select the 3. Condition tab.
2. Click the checkbox in the first row.
3. From the point selector, expand the `DataSim` data domain and select the point `UpdateFrequency`.
4. Click the + button in the left column. The text `$DataSim:UpdateFrequency` should fill in the box.
5. From the drop-down box, choose the > operator.
6. In the right column, enter the number `100`. Your screen should now look like this:



You have set the condition. The expression at the bottom shows what will be passed to Gamma, the internal scripting engine of VINE.

7. Go down to the Configured Actions box and click the Create button.



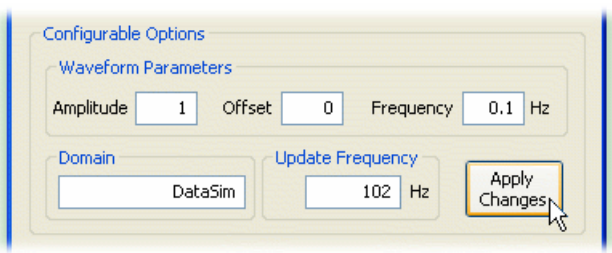
A new configured action should appear in the list. This is a summary of what you have done. When a configured action is selected in this list, you can make changes in any of the tabs and modify it using the Modify. You can also duplicate a configured action using the Create button, or remove it with the Remove button. For more information about configured actions, please refer to Section 10.7, *Configured Actions*

8. Click the Apply button to activate the configured action. Now let's see how it all works.

Trigger and Send an Email

The action you just configured causes VINE to send an email any time the DataSim Update Frequency is changed to a value greater than 100. To test the script, you'll need to trigger it by changing that value in the DataSim.

1. In DataSim, press the More... button to view the Configurable Options
2. Change the Update Frequency to a number greater than `100` and click the Apply Changes button to commit the change.



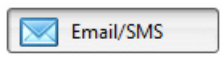
3. Check the email account of the recipient. You should have received a message that looks like this:

Each time you enter a new Update Frequency value greater than `100` in the DataSim, the VINE script will send a similar message.

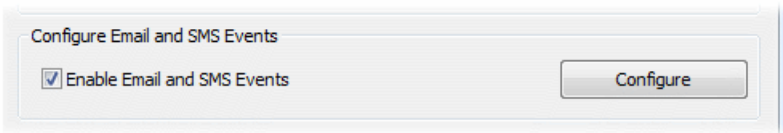
You have just configured and tested an action that sends an email with the name, value, and timestamp of the `Sine` point in the `DataSim` data domain whenever the value of the `UpdateFrequency` point changes to a value over 100. Now you can configure other emails to send your own text messages or HTML pages. The remaining sections in this chapter explain the interface in more detail

10.4. Defining the Email Message

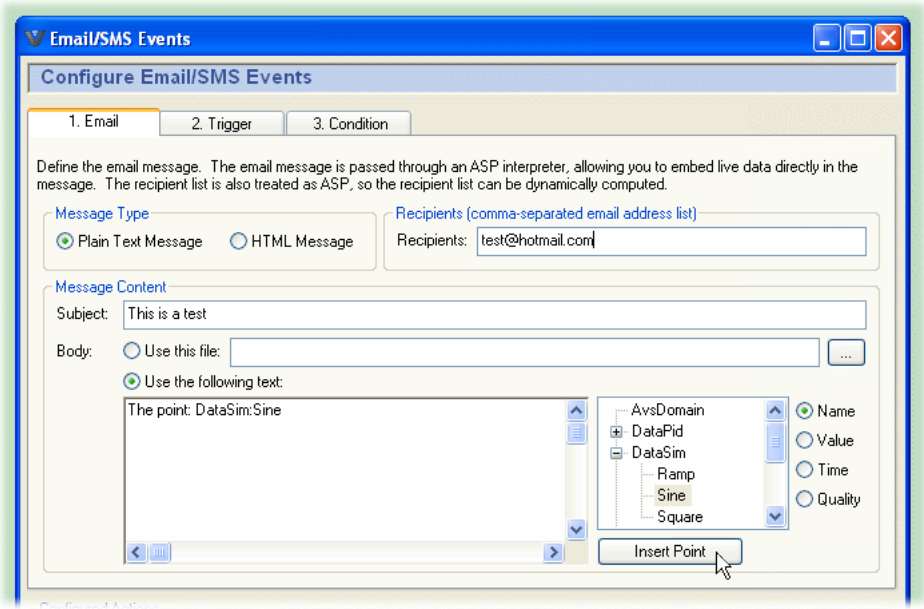
To send an email you need to determine the type of email message, its recipients, title, and message body. This is done from the Email/SMS Events window, which you can access in this way:

1. In the Skkynet VINE Properties window, select Email/SMS 

.
2. In the Configure Email section press the Configure button.



3. Select the 1. Email tab and configure your email as explained below.



Message Type

- **Plain Text Message** sends the text of the message as written in the source file or entered in this interface. Data point values will be assigned at the time the message is sent.
- **HTML Message** sends the source file or entry in this interface as an HTML file. Data point values will be assigned at the time the message is sent.

Recipients

This can be a single email address, or a list of email addresses where each address is separated by a comma. Addresses can be in either of these two forms:

- `username@datadomain.com`
- `User Name <username@datadomain.com>`

It is also possible to create a dynamic list of recipients, as explained in Section 10.9, *Dynamically Changing Email Subjects and Recipients*.

Message Content

- **Subject** Enter the subject of the message.
- **Body** You can use a message from a file, or compose one in the editing box.
  - **Use this file:** lets you insert the name of a file that you want to send as the text of your email. This is not an attachment, but rather the body of your message. Press the … button to browse for the file you need. To see some HTML file examples, please refer to Section 10.8, *HTML Message Examples*.
  - **Use the following text:** lets you write and edit the body of your message. To insert the name, value, timestamp, or quality of the point in the point-picker list, select Name, Value, Time, or Quality button as desired. Then click the Insert Point button. VINE will insert into your text the point name with the proper syntax for the desired output in the email.

If you want to send a message from a file, you can still use the text editor with its convenient interface to create it. Write up the message in the editor, then copy and save it to a file.

The value, time, and quality attributes of the VINE points are accessed by using a special syntax. This is applied automatically in the text editor when you press the Insert button. For your reference, the syntax is as follows:

Button	Syntax	Example
Name	*domainname*:*pointname*	DataSim:Sine
Value	<%= $*domainname*:*pointname* %>	<%= $DataSim:Sine %>
Time	<%= PointTimeString (#$*domainname*:*pointname*) %>	<%= PointTimeString (#$DataSim:Sine) %>
Quality	<%= PointQualityString (#$*domainname*:*pointname*) %>	<%= PointQualityString (#$DataSim:Sine) %>

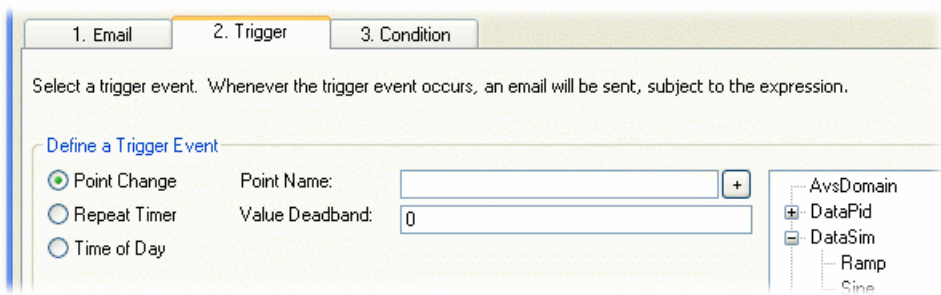
In this syntax, the special characters are used as follows:

Character	Use

Character	Use
`<% ... %>`	The enclosed expression will be evaluated by Gamma, VINE scripting language.
`$`	Indicates to Gamma that this is a VINE point name.
`PointTimeString()`	A Gamma function that returns the timestamp of a VINE point in an easily readable format.
`PointQualityString()`	A Gamma function that returns the quality of a VINE point, as a text string.
#	Protects the VINE point from being evaluated by Gamma until the function is called.

10.5. Assigning a Trigger

A trigger is an event that causes the email to be sent. A trigger event can be either a point value change, a timer event, or a calendar event. You can assign a different trigger for each email, or an identical trigger to any number of emails. An action can be configured to execute on every trigger event, or you can assign trigger conditions that are avaluated whenever a trigger occurs, to determine if the action should be executed.



The three kinds of triggers are:

- **Point Change** fires whenever a specified trigger point changes.

  1. Type the name of the point into the Point Name box, or select the point using the data tree on the right, then click the + button.
  2. (Optional) Enter a value deadband if you want to filter out extraneous data. The number you enter will specify a high and low (plus or minus) range. Any value change falling within that range will not cause the trigger to fire. A positive or negative change greater than this value will activate the trigger and cause the email to be sent.

  To create a trigger that gets reset automatically, please refer to An Auto-Resetting Trigger in Section 10.6, *Setting Trigger Conditions*.

- **Repeat Timer** fires cyclically, each time the number of seconds elapses.
- **Time of Day** fires at the time you specify. You can enter:
  - A number, indicating a specific value. For example, a `0` in the seconds field would cause the event only on the 0th second of the minute. A `30` would indicate only on the 30th second of the minute.
  - A list of numbers, separated by commas. For example, entering `0,15,30,45` in the minutes field would indicate that the event should fire on the hour and at 15, 30 and 45 minutes past the hour.

- A range of numbers, separated by a dash. For example, entering `8-18` in the hours field would indicate that the event should fire every hour from 8 a.m. to 6 p.m.. Ranges can be mingled with lists, as in `0,4,8-16,20`.
- An asterisk (`*`) indicates that the event should fire for every possible value of the field. For example, a `*` in the seconds field would cause the event to fire every second. A `*` in the hours field would cause the event to fire every hour.

To regularly log a record on specific days of the week, please refer to Section 10.6, *Setting Trigger Conditions*.

The ranges of the fields are:

Year:	`1970-*`	Hour:	`0-23`
Month:	`1-12`	Minute:	`0-59`
Day:	`1-31`	Second:	`0-59`

The year and month are entered differently here than for the Gamma `localtime` function, as explained in Time Conditions.

Examples:

- These entries:

  would cause an email to be sent at 8:45 every day, every month, and every year.

- These entries:

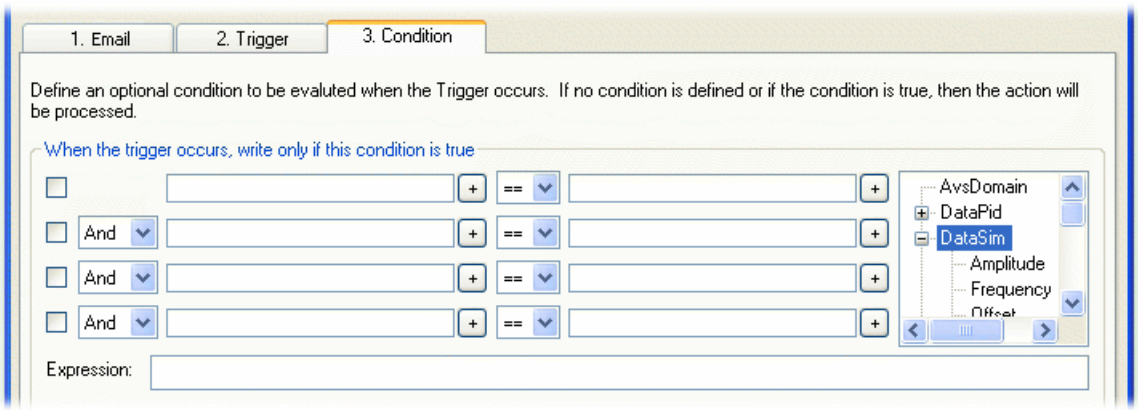
  would cause an email to be sent every hour on the 15th day of each month, every year.

- These entries:

  would cause an email to be sent every second for 5 minutes, every two hours between 8 a.m. and 6 p.m.

10.6. Setting Trigger Conditions

Each action can have up to four conditions that determine whether an email gets sent when the trigger fires.



Fill in the conditions according to the guidelines below. Check the box next to the condition to apply it. As you make entries, the corresponding *Gamma* code will appear in the display. Gamma is VINE's built-in scripting language. The code that appears in the Expression box is the actual code that gets run by Gamma. The order of precedence for "`And`" and "`Or`" operators (`&&` and `||`) is first `And`, then `Or`.

Point Value Conditions

Point names can be entered on either or both sides of the comparison. They can be picked from the data tree list, or typed in. Each point name needs to have a dollar sign (`$`) in front of it to indicate to Gamma that this is a VINE point. You can put numerical values into either side of the comparison.

When you enter a point name in a condition field, the current value of the point will be used in the evaluation. For example, you could define a condition that states that whenever the trigger event occurs, the action will only be executed if another point value is within a certain range.

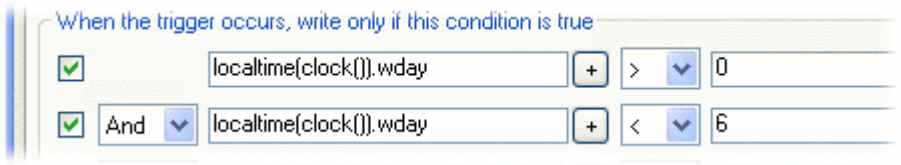
There are three automatic variables available for working with point values:

- `lasttrigger` - the value of the trigger point the last time this trigger was fired.
- `thistrigger` - the value of the trigger point now.
- `lastevent` - the value of the trigger the last time the event was actually executed.

Time Conditions

This provides an additional way to restrict the time, day, month, etc. when a message gets sent. In addition to the options on the triggers, here you have day-of-week condition statements which can give you more flexibility for evemts based on specific days of the week. These will work with any type of trigger event.

You can use the Gamma functions `clock` and `localtime` to specify particular days of the week. For example, these entries:



would create this Gamma code::

```
(localtime(clock()).wday > 0 && localtime(clock()).wday < 6)
```

which would cause an email to be sent only Monday through Friday. The function `localtime` returns a class whose members contain information about the date, as follows:

`.sec`	The number of seconds after the minute (`0` - `59`).
`.min`	The number of minutes after the hour (`0` - `59`).
`.hour`	The number of hours past midnight (`0` - `23`).
`.mday`	The day of the month (`1` - `31`).
`.mon`	The number of months since January (`0` - `11`)
`.year`	The number of years since 1900.
`.wday`	The number of days since Sunday (`0` - `6`).
`.yday`	The number of days since January 1 (`0` - `365`)
`.isdst`	`1` if daylight saving time is in effect, `0` if not, and a negative number if the information is not available.

The year and month are entered differently here than for Time of Day trigger conditions, as explained in Section 10.5, *Assigning a Trigger*.

There are two automatic variables available for working with time values:

- `lasteventtime` - the time that the last event was executed, in UNIX epoch time.
- `curtime` - the UNIX epoch time now.

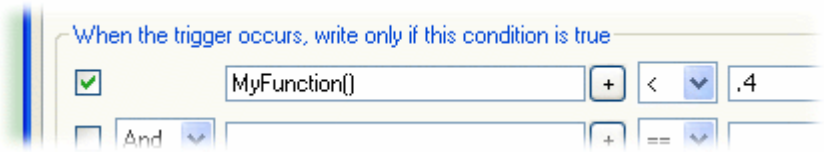
Custom Conditions

If the conditions you need to meet are beyond the scope of this interface, you can use a Gamma function to express virtually any condition you need. Then you can insert the function into one of the condition boxes, and set a condition based on the return value of the function.

To do this is you can create a VINE script (`.g` file) that contains only the functions you will be using for conditions, without any classes or methods. For example, here is the complete contents of such a file, named `MyConditions.g`:

```
function MyFunction ()
{
    myvalue = $DataSim:Sine;
    princ("Value when the trigger fired: ", myvalue, "\n");
    myvalue;
}
```

This function prints the value of the `DataSim:Sine` point, and returns its value. We can use this function as a condition by calling it from one of the condition boxes in the interface, like this:



When the trigger fires, `MyFunction` is called, and the return value gets checked to see if it is less than .4. If so, the email is sent.

An Auto-Resetting Trigger

This script can turn any VINE point into a trigger that automatically resets. To use it, you first need to load and run the `TriggerFunctions.g` script (shown below and included in the installation archive). Then, if you put this formula:

```
HighWithReset($default:TriggerPoint)!= nil
```

into the condition boxes, whenever the *`TriggerPoint`* changes to a non-zero number in VINE, your trigger will fire. The script waits for a millisecond, then resets the *`TriggerPoint`* back to zero. The second function works similarly, but triggers on a change to zero, instead of a change to a non-zero number.

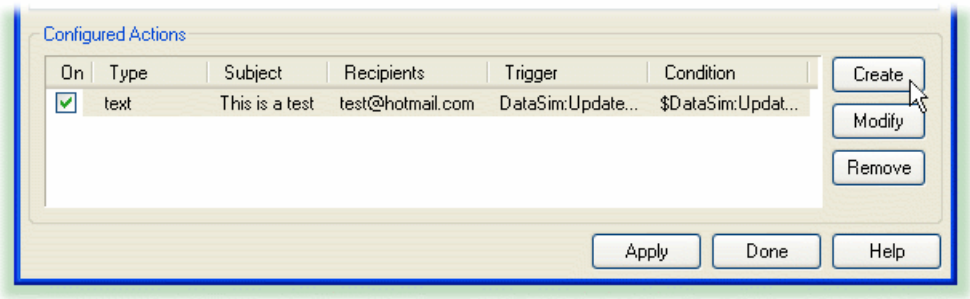
TriggerFunctions.g

```
/*
 * This file contains handy functions to perform more complex condition
 * handling in the Condition tab of the data logging and email interfaces.
 */

/*
 * Test a trigger point for a non-zero value.  If the point is non-zero,
 * create a delayed event to reset the point to zero, and return true,
 * indicating that the condition has succeeded and the action should proceed.
 * If the value is 0, then simply return nil indicating that the action
 * should not proceed.  We need to test for zero because when we reset the
 * trigger point to zero a second data change event will occur.
 *
 * The argument is unevaluated, so the condition should look like this:
 *    HighWithReset($default:TriggerPoint) != nil
 */

function HighWithReset(!triggerpoint)
{
    local   value;
    if (!undefined_p(value = eval(triggerpoint)) && value != 0)
    {
        after(0.001, 'setq(@triggerpoint, 0));
        t;
    }
    else
    {
        nil;
    }
}

/*
 * This is the inverse of HighWithReset (see above).  If the trigger point
 * is zero, perform the action and set the trigger point to 1.  If the
 * trigger point is non-zero do nothing and return nil.
 */
function LowWithSet(!triggerpoint)
{
    local   value;
    if (!undefined_p(value = eval(triggerpoint)) && value == 0)
    {
```

```
        after(0.001, 'setq(@triggerpoint, 1));
        t;
    }
    else
    {
        nil;
    }
}
```

10.7. Configured Actions

A *configured action* will cause a given email to be sent, based on a trigger and optional conditions. It is the end result of your configuration activities in this interface. The Configured Actions list shows the actions you have configured, and allows you to create, modify, or remove actions, as well as turn them on or off.



The list of configured actions shows the actions you have already configured. Selecting an existing action from the list automatically fills in the Email, Trigger, and Condition tabs with its information. Checking or unchecking the On box at the left lets you switch the action on or off.

The Create button creates an action for the information currently entered in the Email, Trigger, and Condition tabs. If you press the Create button while a configured action is selected, it creates a duplicate of that configured action and adds it to the list. This is a quick way to configure similar actions.

The Modify button overwrites the selected configured action with the information currently entered in the Email, Trigger, and Condition tabs.

The Remove button removes a configured action.

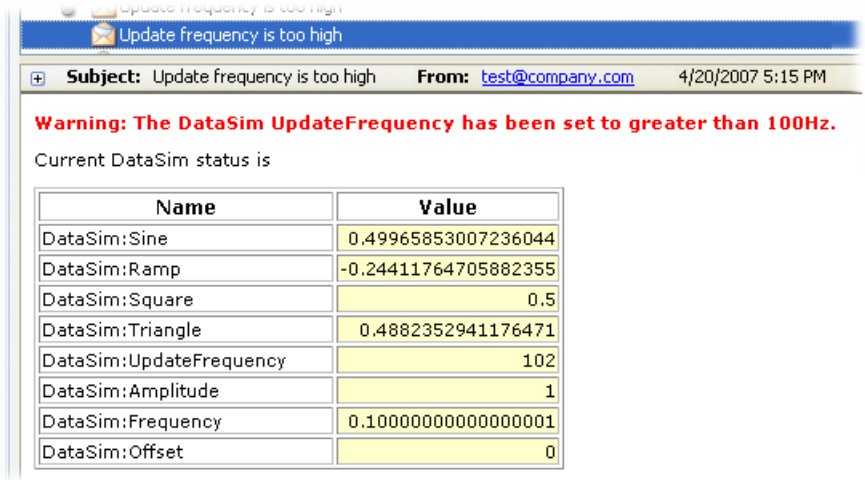
Once a configured action has been created or modified, the changes won't take effect until you click the Apply or Done button.

10.8. HTML Message Examples

Sending an HTML message is as simple as clicking the HTML Message button in the Email tab. For the Message Source you can choose a file or write HTML code directly into the text-entry box. Here are two examples of how you can embed data into an HTML messages, using an ASP source file

10.8.1. An HTML Message with Embedded Data Points

This is an example of an ASP file that embeds the latest data from VINE points into an HTML table. The ASP file is named `EmbedPoints.asp`, and comes installed with VINE in the `Skkynet VINE/etc/` directory. If VINE is configured to send this file as the message body, and the DataSim's UpdateFrequency is changed to, say, `102`, VINE will email a message like this:



Contents of the ASP File

The contents of the file `EmbedPoints.asp` are as follows:

```
<html>
<style>
BODY, P, TD{
background-color : White;
font-family :  Verdana, Geneva, Arial, Helvetica, sans-serif;
font-size : 8pt;
}
TH{
font-family:Verdana, Arial, Helvetica, sans-serif;
font-size: 9pt;
font-weight: bold;
background-color: #23cce6fe;
}
.highlight{background-color: #FFFFCC; text-align:right;}
.warning{color: #FF0000; font-weight: bold;}
</style>
<body>
<!--
 This is a simple example of an HTML template file which contains
  embedded point values from the DataHub.
 -->
<p></p>
<div class="warning">Warning: The DataSim UpdateFrequency has been
                    set to greater than 100Hz.</div>
<p></p>
Current DataSim status is
<p></p>
<table border="1">
  <tr>
    <th width="180">Name</th>
    <th width="80">Value</th>
  </tr>
  <tr>
    <td>DataSim:Sine</td>
    <td class="highlight"><%=$DataSim:Sine%></td>
  </tr>
  <tr>
    <td>DataSim:Ramp</td>
    <td class="highlight"><%=$DataSim:Ramp%></td>
  </tr>
  <tr>
    <td>DataSim:Square</td>
    <td class="highlight"><%=$DataSim:Square%></td>
  </tr>
  <tr>
```

```
    <td>DataSim:Triangle</td>
    <td class="highlight"><%=$DataSim:Triangle%></td>
  </tr>
  <tr>
    <td>DataSim:UpdateFrequency</td>
    <td class="highlight"><%=$DataSim:UpdateFrequency%></td>
  </tr>
  <tr>
    <td>DataSim:Amplitude</td>
    <td class="highlight"><%=$DataSim:Amplitude%></td>
  </tr>
  <tr>
    <td>DataSim:Frequency</td>
    <td class="highlight"><%=$DataSim:Frequency%></td>
  </tr>
  <tr>
    <td>DataSim:Offset</td>
    <td class="highlight"><%=$DataSim:Offset%></td>
  </tr>
</table>
</body>
</html>
```

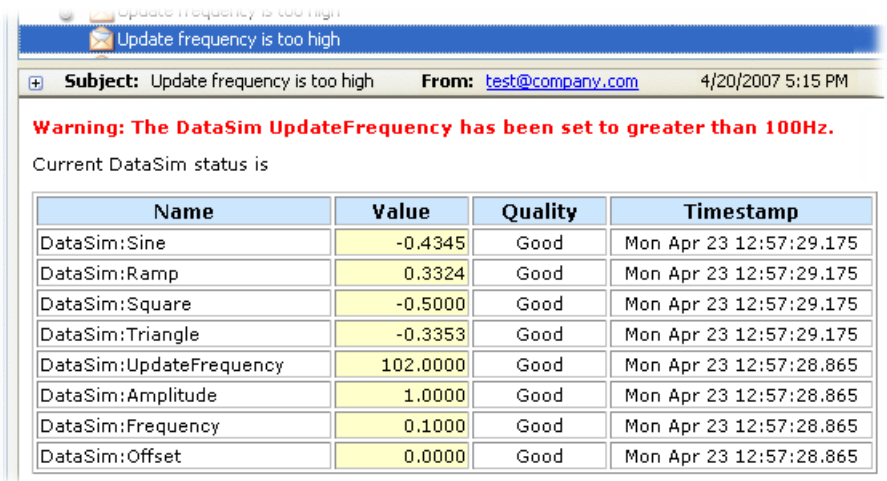
This file consists of HTML code interspersed with *Gamma* code. Gamma is the scripting language of Skkynet VINE. The Gamma code is often used to determine the value of a VINE point, with the following syntax:

```
<%=$domainname:pointname%>
```

The pointed brackets and percent signs (`<% ... %>`) indicate to the VINE ASP interpreter that this is Gamma code. The equals sign (`=`) tells Gamma to evaluate the expression, and the dollar sign (`$`) tells Gamma that this is a VINE point.

10.8.2. An HTML Message with a Table Created in Code

This is an example of an HTML message with a table created by using code, rather than explicitly writing it out. Using code provides more flexibility in formatting the data and making changes to the table. The code is written into an ASP file named `CreateTable.asp`, which comes installed with VINE in the `Skkynet VINE/etc/` directory. If VINE is configured to send this file as the message body, and the DataSim's UpdateFrequency is changed to, say, `102`, VINE will email this message:



Contents of the ASP File

The contents of the file `CreateTable.asp` are as follows:

```
<html>
<style>
BODY, P, TD{
    background-color : White;
    font-family :  Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size : 8pt;
}
TH{
    font-family:Verdana, Arial, Helvetica, sans-serif;
    font-size: 9pt;
    font-weight: bold;
    background-color: #cce6fe;
}
.highlight{background-color:#FFFFCC; text-align:right}
.warning{color: #FF0000; font-weight: bold;}
</style>
<body>
<p></p>
<div class="warning">Warning: The DataSim UpdateFrequency has been set
to greater than 100Hz.</div>
<p></p>
Current DataSim status is
<p></p>
<table border="1">
    <tr>
    <th width="180">Name</th><th width="80">Value</th>
    <th width="80">Quality</th><th width="160">Timestamp</th>
    </tr>
<%
require ("Time");
require ("Quality");

try
{
    local  v, q, tm, ts, info;

    with pt in [ #$DataSim:Sine, #$DataSim:Ramp, #$DataSim:Square,
                 #$DataSim:Triangle, #$DataSim:UpdateFrequency,
                 #$DataSim:Amplitude, #$DataSim:Frequency, #$DataSim:Offset ] do
    {
        info = PointMetadata (pt);
        v = eval (pt);
        if (!number_p(v)) v = 0;
        q = GetQualityName (info.quality);
        ts = PointGetUnixTime (pt);
        tm = format ("%19.19s.%03d", date(ts), (ts % 1.0) * 1000);
%>
        <tr><td><%= pt %></td>
            <td class="highlight"><%= format("%.4f",v) %></td>
            <td align="center"><%= q %></td>
            <td align="center"><%= tm %></td></tr>
<%
    }
}
catch
{
    princ (_last_error_, "\n");
    print_stack (nil, _error_stack_);
}
%>
</table>
</body>
</html>
```

This file consists of HTML code interspersed with *Gamma* code. Gamma is the scripting language of Skkynet VINE. The Gamma code is often used to determine the value of a VINE point, with the following syntax:

```
<%=$domainname:pointname%>
```

The pointed brackets and percent signs (`<% ... %>`) indicate to VINE that this is Gamma code. The equals sign (`=`) tells Gamma to evaluate the expression, and the dollar sign (`$`) tells Gamma that this is a VINE point. Other Gamma statements and functions used in this example include `require`, `try`, `local`, `with`, `if`, `format`, `catch`, `princ`, and `print_stack`. The functions `GetQualityName` and `PointGetUnixTime` are from the required files `Quality.g` and `Time.g` respectively.

10.9. Dynamically Changing Email Subjects and Recipients

The ASP processor in Gamma allows you to embed the result of any Gamma expression within the subject and recipient fields of an email. To do this on the subject field, you would use the same `<%= %>` syntax as is available for messages, for example:

```
The Sine value is now <%= $DataSim:Sine %>
```

would put the value of the `DataSim:Sine` point into the subject line of the email or message.

This syntax, explained in a the end of Section 10.4, *Defining the Email Message* can also be used to insert addresses for one or more the message recipients, by creating a point that contains the list of recipient names. The value of this point could then be changed externally based on who is on-call or is logged into an attached SCADA system. For example, a point in the `default` domain named `CurrentOperatorEmail`, would be entered in the Recipients: field like this:

```
<%= $default:CurrentOperatorEmail %>
```

If you need a more complex calculation to determine the recipients, you could create a Gamma script that loads when VINE starts. For example, to change the email based on the value of a point, you could do something like this:

```
function choose_mail_recipient()
{
    if ($DataSim:Sine > 0.5)
        "operator1@gmail.com";
    else
        "operator2@gmail.com";
}
```

and then put the appropriate function call into the email recipient list, like this:

```
<%= choose_mail_recipient() %>
```

Notice that the expression within `<%= %>` does not end with a semicolon. This syntax requires a Gamma *expression*, not a Gamma *statement*. Effectively, it needs to be code that would be syntactically correct in this statement:

```
x = insert_expression_here;
```

Chapter 11. VINE Scripting

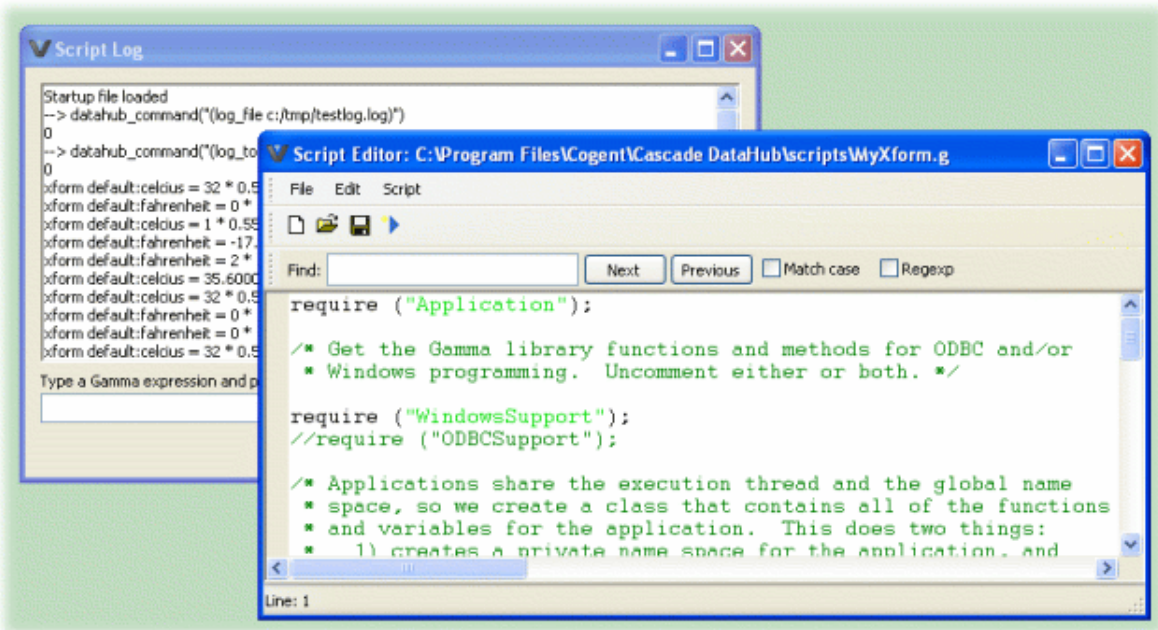
VINE has a powerful, built-in scripting language called *Gamma*. Using Gamma, you can write scripts to interact with VINE and its data in various ways, such as:

- Attach scripts to specific data points so the scripts are run whenever the point value changes.
- Build custom dashboards and summary displays directly in Gamma scripts to create self contained VINE applications.
- Create alarm condition scripts and have them display warning messages to the user.
- Create Excel readable log files from your live data by running logging code on a timed interval, or whenever a point change occurs.
- Connect to ODBC compliant relational databases to extract data as well as create records from live data.
- Apply linear transforms on data as it passes through the DataHub (for example change a temperature reading from Celsius to Fahrenheit).
- Create full simulation programs to test production systems before you 'go live'.

Please refer to the VINE Scripting manual for more information about scripting.

11.1. Tools

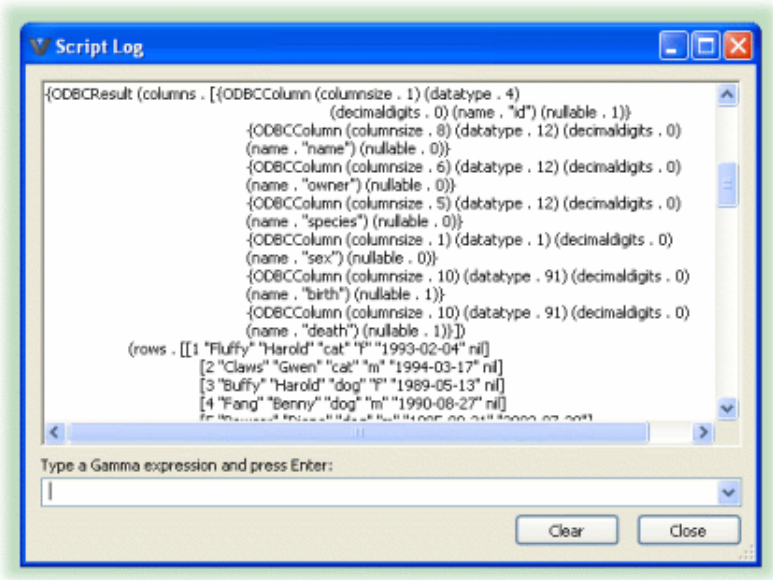
VINE comes with a built-in Script Editor for writing and editing scripts, as well as a Script Log for viewing script outputs.


Script Log
Startup file loaded
--> datahub_command("(log_file c:/tmp/testlog.log)")
0
Type a Gamma expression and p
Script Editor: C:\Program Files\Cogent\Cascade DataHub\scripts\MyXform.g
File Edit Script
Find: Next Previous Match case Regexp
require ("Application");
/* Get the Gamma library functions and methods for ODBC and/or
* Windows programming. Uncomment either or both. */
require ("WindowsSupport");
//require ("ODBCSupport");
/* Applications share the execution thread and the global name
* space, so we create a class that contains all of the functions
* and variables for the application. This does two things:
Line: 1

Please refer to the VINE Scripting manual for more information about how to use these tools.

11.2. VINE ODBC (Open Database Connectivity) Scripting

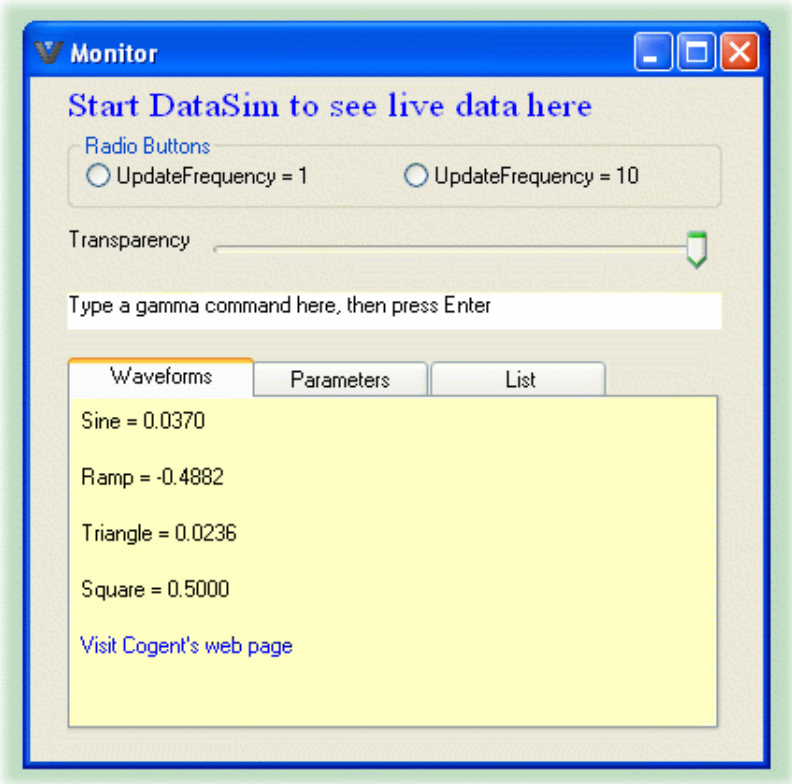
The ODBC support in VINE provides an interface to any ODBC-compliant database. It lets you create a class for any database table, and assign each column of the table as an instance variable of the class, giving you complete access to any point of data in the database.


Script Log
{ODBCResult (columns . [{ODBCColumn (columnsize . 1) (datatype . 4)
(decimaldigits . 0) (name . "id") (nullable . 1)}
{ODBCColumn (columnsize . 8) (datatype . 12) (decimaldigits . 0)
(name . "name") (nullable . 0)}
{ODBCColumn (columnsize . 6) (datatype . 12) (decimaldigits . 0)
(name . "owner") (nullable . 0)}
{ODBCColumn (columnsize . 5) (datatype . 12) (decimaldigits . 0)
(name . "species") (nullable . 0)}
{ODBCColumn (columnsize . 1) (datatype . 1) (decimaldigits . 0)
(name . "sex") (nullable . 0)}
{ODBCColumn (columnsize . 10) (datatype . 91) (decimaldigits . 0)
(name . "birth") (nullable . 1)}
{ODBCColumn (columnsize . 10) (datatype . 91) (decimaldigits . 0)
(name . "death") (nullable . 1)}]
(rows . [[1 "Fluffy" "Harold" "cat" "f" "1993-02-04" nil]
[2 "Claws" "Gwen" "cat" "m" "1994-03-17" nil]
[3 "Buffy" "Harold" "dog" "f" "1989-05-13" nil]
[4 "Fang" "Benny" "dog" "m" "1990-08-27" nil]
Type a Gamma expression and press Enter:
Clear
Close

Please refer to the VINE ODBC Support manual for more information.

11.3. VINE Windows Scripting

VINE offers Windows scripting support, with the classes necessary to create windows, buttons, frames, tabs, entry fields, and so on—all animated with live data. Here is a screenshot of a test program:



Please refer to the VINE Windows Scripting manual for more information. The code for this example is in the `WindowsExample.g` file included in your distribution.

Chapter 12. Security

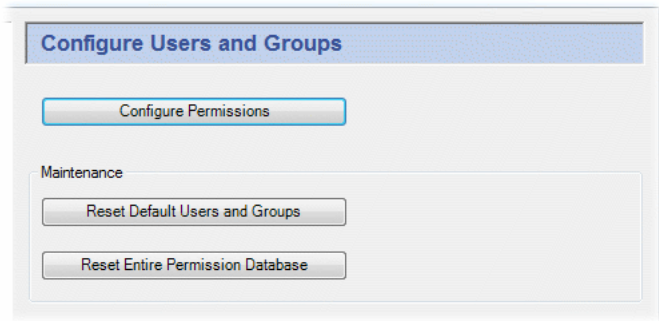
Skkynet VINE provides a means for full access control to all DDE, TCP, OPC, and tunnel/mirror connections, using authentication and authorization. *Authentication* limits access to recognized users, based on a username/password combination. *Authorization* provides a set of permissions for each user, allowing access to certain functions while denying access to others.

VINE also provides full SSL (Secure Sockets Layer) encryption for TCP/IP tunnelling and mirroring connections.

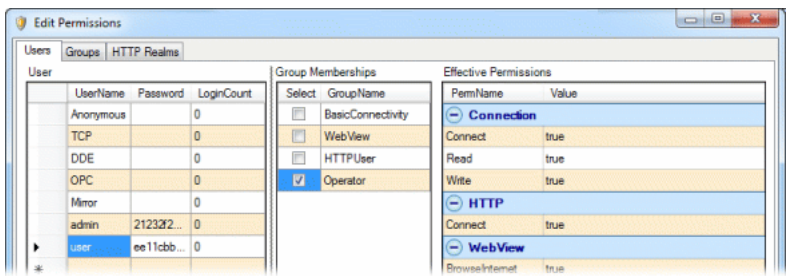
12.1. How to Configure

In the VINE Properties window, select Security

.



Click the Configure Permissions button to open the Edit Permissions window.

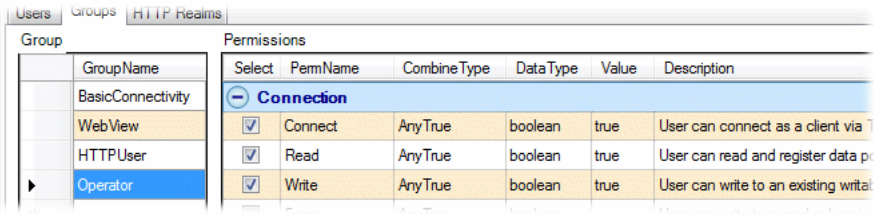


Here you can create and modify groups, and then assign users to those groups.

Groups

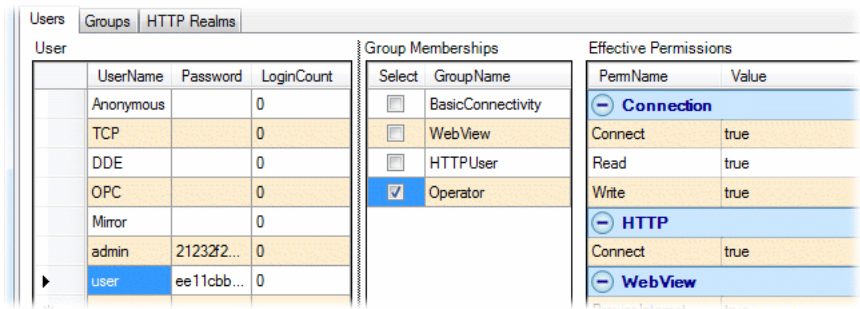
Groups provide a convenient way to configure a number of users who have identical permissions. Each group can be assigned a unique set of permssions from the Permissions table. There are three default groups: `Permissive`, `WebView`, and `HTTPUser`. To add a group, type a group name in the bottom row of the Groups table. Check or uncheck the boxes to assign permissions.

For example, in the illustration below an `Operator` has been added that has been given Connection permissions for `Connect`, `Read`, and `Write`.

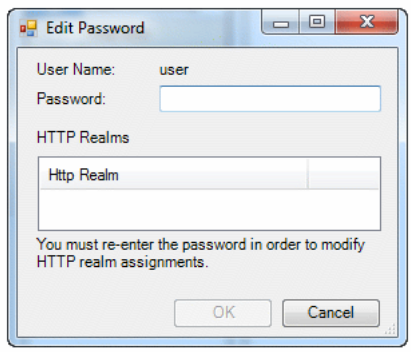


Users

There are two kinds of users--normal and special. Normal users correspond to individuals with a name and a password. Special users provide a way to offer different security models for different protocols. For more information on types of users, please refer to Section 12.3, *User Authentication*.



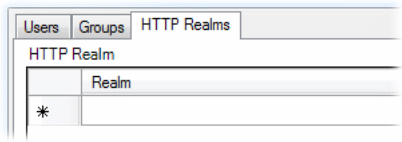
To add a user, type a user name in the bottom row of the User table. When you press **Enter**, a password dialog will appear:



Enter a password and select an HTTP realm for that user. When you click OK, a string of charaters will appear in the Password field for that user. Passwords are stored using a reasonably strong non-reversible encryption. If a user forgets his password, it is not recoverable. To change HTTP realm for a user, their password must be reentered. For more information on passwords, please refer to Section 12.6, *Passwords*.

HTTP Realms

Here you can maintain a list of HTTP authentication realms. This list is accessed by the VINE Web Server, as described here: .



To add an authentication realm, simply type it into the list. One or more of these realms are assigned to each user when their password is configured (see above).

Common Scenario

The most common Skkynet VINE security configuration is to allow any user to connect via OPC or DDE, while only allowing authorized users to connect via TCP or via a tunnel/mirror. This eliminates exposure of the the TCP and tunnel/mirror connections to unwanted Internet and network clients. OPC and DDE are not exposed in this way.

To configure this scenario, you need to remove all group memberships from the special `Anonymous`, `TCP`, and `Mirror` users. Simply click on each of these user names in turn, and uncheck all group memberships for that user. When you are finished, only `DDE` and `OPC` should have any group memberships.

12.2. SSL and Firewalls

Skkynet VINE provides the option to use SSL encryption to protect your data when tunnelling/mirroring to another VINE across a network connection. The SSL implementation uses the default SSL-3 encryption cipher: DHE-RSA-AES256-SHA, which is a 256-bit encryption method. The Tunnel/Mirror section of this manual explains how to configure SSL for tunnelling and mirroring.

SSL Certificates

An SSL certificate is required to use SSL encryption between VINEs. VINE installs a default SSL Certificate for you, but you can use your own certificate if you prefer.

Because VINE is often used under circumstances where it is not possible or desirable to connect to the Internet, it does not check the issuing authority for a security certificate against the IP address or DNS name of the target computer, nor does it check the expiry date of the certificate. In most control applications it is not acceptable for VINE to refuse access to a critical process due to a simple mismatch in machine name declaration or the expiry of a time-limited certificate. Thus, any well-formed certificate will be accepted. Most important, the data encryption will be performed regardless of the validity of the certificate.

Firewall Ports

VINE lets you specify which ports it will use for tunnelling/mirroring over a network. Firewalled ports can be secured, because if you open a port on the firewall, any program that attempts to connect on this port will need to be able to communicate with VINE that is listening on that port. As long as authentication is used for tunnelling, even a user who attempts to connect using another VINE program will need to have access to a valid username and password.

12.3. User Authentication

Authentication of users is applied on a per-connection basis. This means that whenever a client program connects to Skkynet VINE, it must transmit a user name and password in order to authenticate. Until the

client program authenticates, it operates with the permissions of the anonymous user (see below). After 5 seconds, the permissions currently in force for the client are checked for the `Connect` permission. If the client does not have `Connect` permission, the connection is terminated. The client may authenticate as another user at any time after it has connected. If a client transmits an incorrect user name or password, it is not immediately disconnected, but instead keeps the permissions the were in force prior to the authentication attempt.

Special Users

To facilitate special connectivity needs, Skkynet VINE has several special users. The *anonymous user* represents a client that has not authenticated. When a client first connects, it is given the permissions of the anonymous user. The client may continue to operate with the anonymous user permissions (so long as the anonymous user has the `Connections Connect` permission), or may authenticate as another user at any time. In essence, the security of VINE is no greater than the permissions given to the anonymous user. The default distribution of VINE has anonymous user permissions enabled.

In addition to the anonymous user, there are special users associated with each connection protocol. These are essentially anonymous users that are associated with just one particular protocol. The protocols are:

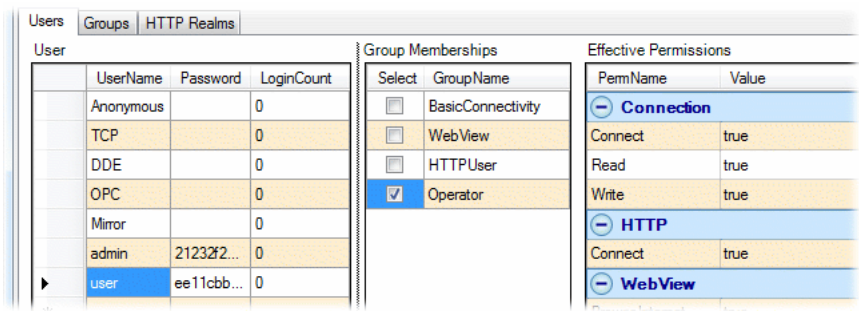
DDE:	Any connection made from a DDE client to Skkynet VINE.
OPC:	Any connection made from an OPC client to Skkynet VINE.
TCP:	Any connection made from a third-party program using a direct TCP connection, the VINE API, or a Java applet embedded in a web browser.
Mirror:	A mirror or tunnel connection from another Skkynet VINE

When a client connects using one of the above protocols, it is originally given the anonymous user permissions, and then promoted to the protocol user associated with the connection type, once the connection is fully constructed. This allows Skkynet VINE to apply different permissions to anonymous connections of different types.

OPC users: Since the OPC protocol does not provide a mechanism for authentication, this is the only mechanism available to limit the permissions of an OPC client.

Normal Users and Groups

Clients to Skkynet VINE are referred to as *users*. A user name is any combination of letters, numbers and some punctuation characters. A password can be any sequence of characters. Each user has an associated set of permissions. When a client transmits a correct user name and password, it acquires the permissions of that user.



Users can be assigned to a *group* to simplify the configuration of many users who have identical permissions. A user can be added to a group at any time. When added to a group, the user's permissions

will be altered to match those of the group. If the permissions for the group are subsequently changed, the change will immediately affect all users in the group. A user may only belong to one group at a time.

12.4. Authorization and User Permissions

Every client program connected to Skkynet VINE is associated with exactly one user at any given time. Each user is authorized to access certain features of VINE according to its *user permissions*. When a client first connects, it is immediately associated with the anonymous user, and gets those permissions. Then it gets switched to the special user for the protocol it is connecting on. If the client subsequently authenticates itself as a normal user, it is then granted that user's permissions. A client's permissions are always the entire permission set for the user that it is currently associated with.

To edit user permissions select the user name in the Users list and press the Add button. This will open the Permission Editor.

Permissions are categorized into four groups, and defined as follows:

Connection

Connection Connect

This user is allowed to maintain a connection to Skkynet VINE. When a connection is made, the client has a 5-second grace period in which to authenticate before the client is disconnected. If the client does not have Connection Connect permissions after the grace period expires, it will be disconnected.

Connection Read

This user is allowed to read point values and subscribe to point value changes.

Connection Write

This user is allowed to write a new point value to Skkynet VINE.

Connection Force

If the user has Connection Write permission, he may also have this permission. In this case, the user will able to send the **force** and **cforce** commands to VINE, which will override the read-only status and timestamp check for a point, thereby forcing a write to succeed where it would otherwise fail.

Connection CreatePoint

This user is allowed to create new points in existing data domains in Skkynet VINE.

Connection DeletePoint

This user is allowed to delete a point from Skkynet VINE.

Normally, no client should be allowed to delete points from Skkynet VINE. Deleting points can be very disruptive for existing clients. Use this permission with caution.

Connection CreateDomain

This user is allowed to create new data domains. Normally you should also set Connection CreatePoint permission when you set this permission for a user.

Connection ChangeModel

This user can change the organizational hierarchy of the data points. This is the information that makes the data appear as a tree structure in the Data Browser.

Connection LoadConfig

This user is allowed to tell Skkynet VINE to load a specific configuration file.

Connection Connection UserAdmin

This user is allowed to create and edit users and groups non-interactively. This permission is not currently available.

Connection Connection ConfigAdmin

This user is allowed to transmit commands to Skkynet VINE to alter VINE's configuration. This normally includes actions like enabling and disabling particular interfaces and functions within VINE.

Connection Connection Shutdown

This user may transmit an **exit** command to Skkynet VINE, causing it to shut down.

Restrictions

Restrictions LimitConcurrentLogins

This user will be limited to the number of concurrent connections specified in Restrictions MaxLoginLimit, regardless of the connection type. For example, if the Restrictions MaxLoginLimit is 2, the user would be allowed to make two TCP connections, or one TCP and one DDE connection. This option also applies to anonymous users. This restriction is not currently being enforced.

Restrictions LimitTotalLogins

This user is allowed to connect to Skkynet VINE at most the number of times specified by Restrictions MaxLoginLimit. Once the user has connected to VINE this many times, future attempts to log in will be refused, even after VINE has been restarted. This restriction is not currently being enforced.

Restrictions Expires

This user will be allowed to log in to Skkynet VINE up to, but not including, the date specified by the Restrictions ExpiryDate. This restriction is not currently being enforced.

Restrictions MaxLoginLimit

An integer specifying the maximum total number of logins permitted, used by Restrictions LimitTotalLogins. This restriction is not currently being enforced.

Restrictions ConcurrentLoginLimit

An integer specifying the maximum number of concurrent logins, used by Restrictions LimitConcurrentLogins. This restriction is not currently being enforced.

Restrictions ExpiryDate

A date, used by Restrictions Expires. This restriction is not currently being enforced.

HTTP

HTTP Connect

This user is allowed to connect to the VINE Web Server.

WebView

WebView Connect

This user is allowed to connect to WebView. When a connection is made, the client has a 5-second grace period in which to authenticate before the client is disconnected. If the client does not have WebView Connect permissions after the grace period expires, it will be disconnected.

WebView CreatePage

This user is allowed to create pages in WebView.

WebView CreateControl

This user is allowed to create controls in WebView. This function is not currently available.

WebView SavePage

This user is allowed to save his own pages in WebView.

WebView SaveControl

This user is allowed to save controls in WebView. This function is not currently available.

WebView DeletePage

This user is allowed to delete his own pages in WebView. This function is not currently available.

WebView DeleteControl

This user is allowed to delete controls in WebView. This function is not currently available.

WebView ViewPage

This user is allowed to view his own pages in WebView.

WebView EditPage

This user is allowed to edit his own pages in WebView.

WebView ViewOtherOwnerPage

This user is allowed to view pages belonging to other owners in WebView.

WebView EditOtherOwnerPage

This user is allowed to edit pages belonging to other owners in WebView.

WebView SaveOtherOwnerPage

This user is allowed to save pages belonging to other owners in WebView.

WebView DeleteOtherOwnerPage

This user is allowed to delete pages belonging to other owners in WebView. This function is not currently available.

WebView EditOtherOwnerControl

This user is allowed to edit controls belonging to other owners in WebView. This function is not currently available.

WebView SaveOtherOwnerControl

This user is allowed to save controls belonging to other owners in WebView. This function is not currently available.

WebView DeleteOtherOwnerControl

This user is allowed to delete controls belonging to other owners in WebView. This function is not currently available.

WebView ViewOnlineHelp

This user is allowed to view the WebView online help.

WebView BrowseInternet

This user is allowed to browse the Internet from within WebView.

WebView ChangeTheme

This user is allowed to change the WebView theme. This function is not currently available.

WebView ChangeOptions

This user is allowed to change WebView options.

WebView Troubleshoot

This user is allowed to use the troubleshooting abilities of WebView.

WebView ConfigureTraceSettings

This user is allowed to configure trace settings in WebView.

12.5. Permissions for the VINE Command Set

Each time VINE receives a command from a client, it checks the client's user permissions. Before executing the command, VINE compares the user's permissions to the permissions required to run the command (shown in the table below). If the user has the necessary permissions, the command is executed, otherwise an error message is returned.

Command Name	Permissions Required
acksuccess	none
add	Connection Write
alive	none
append	Connection Write
assembly	Connection ConfigAdmin
attribute	Connection ConfigAdmin
auth	none
authgroup	Connection ConfigAdmin
authuser	Connection ConfigAdmin
auto_create_domains	Change auto domain creation
auto_timestamp	Connection ConfigAdmin
bandwidth_reduce	none
bridge	Connection ConfigAdmin, Connection Write
bridge_remove	Connection ConfigAdmin
bridge_transform	Connection ConfigAdmin
cforce	Connection Write, Connection Force
cread	Connection Read, Connection CreatePoint
create	Connection CreatePoint
create_domain	Connection CreateDomain
report	Connection Read, Connection CreatePoint
cset	Connection Write, Connection CreatePoint
cwrite	Connection Write, Connection CreatePoint
debug	Connection ConfigAdmin
defaultprop	Connection ConfigAdmin

Command Name	Permissions Required
delete	Connection DeletePoint
deleted	Connection DeletePoint
div	Connection Write
domain	none
drop_license	Connection Connect
dump	Connection ConfigAdmin
echo	Connection Write
enable_bridging	Connection ConfigAdmin
enable_connect_server	Connection ConfigAdmin
enable_dde_client	Connection ConfigAdmin
enable_dde_server	Connection ConfigAdmin
enable_mirror_master	Connection ConfigAdmin
enable_mirror_slave	Connection ConfigAdmin
enable_scripting	Connection ConfigAdmin
enable_tcp_server	Connection ConfigAdmin
error	none
exception_buffer	Connection ConfigAdmin
execute_plugin	Connection ConfigAdmin
exit	Connection Shutdown
failed_license	Connection ConfigAdmin
flush	Connection ConfigAdmin
force	Connection Write, Connection Force
format	Connection Connect
heartbeat	none
ignore	Connection Read
ignore_old_data	Connection ConfigAdmin
include	Connection LoadConfig
instance	Connection ConfigAdmin
load_config_files	Connection LoadConfig
load_plugin	Connection ConfigAdmin
load_scripts	Connection ConfigAdmin
lock	Connection Write
log_file	Connection ConfigAdmin
log_to_file	Connection ConfigAdmin
master_host	Connection ConfigAdmin
master_service	Connection ConfigAdmin
mirror_master	Connection ConfigAdmin
mirror_master_2	Connection ConfigAdmin
mult	Connection Write
on_change	Connection ConfigAdmin
point	Connection Write
private_attribute	Connection ConfigAdmin

Command Name	Permissions Required
property	Connection ConfigAdmin
quality	Connection Write
read	Connection Read
readid	Connection Read
register_datahub	Connection Read
report	Connection Read
report_all	Connection Read
report_domain	Connection Read
report_errors	Connection Read
request	Connection Read
request_initial_data	Connection Read
secure	Connection Write
set	Connection Write
show_data	Connection ConfigAdmin
show_debug_messages	Connection ConfigAdmin
show_event_log	Connection ConfigAdmin
show_icon	Connection ConfigAdmin
show_properties	Connection ConfigAdmin
show_script_log	Connection ConfigAdmin
slave	Connection Read
subassembly	Connection ConfigAdmin
success	none
sync	Connection Write
taskdied	Connection ConfigAdmin
taskstarted	Connection ConfigAdmin
tcp_service	Connection ConfigAdmin
timeout	none
transmit_insignificant	Connection ConfigAdmin
type	Connection ConfigAdmin
unload_plugin	Connection ConfigAdmin
unreport	Connection Read
version	none
warn_of_license_expiry	Connection ConfigAdmin
write	Connection Write
OPC-specific commands	
enable_opc_client	Connection ConfigAdmin
enable_opc_server	Connection ConfigAdmin
OPCAddItem	Connection Write
OPCAttach	Connection ConfigAdmin
OPCDetach	Connection ConfigAdmin
OPCInit	Connection ConfigAdmin
DDE-specific commands	

Command Name	Permissions Required
DDEAdvise	Connection Write
DDEConnect	Connection ConfigAdmin
DDEInit	Connection ConfigAdmin
DDEService	Connection ConfigAdmin
DDEUnadvise	Connection Write
DDEUnadvisePoint	Connection Write
EnableDDEServer	Connection ConfigAdmin

12.6. Passwords

The authentication information for passwords is stored in a database in the configuration directory in a non-reversible encryption. They are secure.and non-recoverable. If a user forgets his password, it cannot be retrieved or regenerated.

When a password is associated with a mirror/tunnel connection, it is stored in a weakly encrypted form on disk, in the `Skkynet VINE.cfg` file. This is a reversible encryption, so a good security policy would be to deny access to this file to untrusted users.

When a password is transmitted across the network, it is transmitted in plain text. This is necessary to accommodate the variety of clients that could generate an authentication request. If the network is itself insecure, it is advisable to use a VPN (Virtual Private Network) or enable SSL for mirror/tunnelling to encrypt the network traffic.

Chapter 13. Working With Data

This chapter gives an overview of how Skkynet VINE handles data and the various protocols it works with.

13.1. Data Points

Each value stored in Skkynet VINE is called a *point*. A point has the following attributes:

- **Name:** a character string. Currently the only limit on length is internal buffer size, about 1000 bytes by default.
- **Value:** an integer, floating-point number, or character string.
- **Time:** the date and time of the last significant change to the point's value, confidence, quality or other status information.
- **Quality:** the quality of the connection, assigned by Skkynet VINE for this point, such as `Good`, `Bad`, `Last known`, `Local override`, etc.
- **Confidence:** a value from `0` to `100` that indicates as a percentage the probability that the value shown for the point is actually its true value. This feature can be accessed and changed only by using the API. Skkynet VINE never uses confidence itself, but carries it for use by client applications.

13.1.1. Creating New Points

Skkynet VINE automatically creates a point whenever a connecting program tries to read, write, or create a point that doesn't exist. When the point is created, Skkynet VINE assigns its value, time, quality, and confidence.

It is possible to have Skkynet VINE create points and assign values to them at startup. Sometimes referred to as *seeding*, this is done with supplemental configuration files. Please refer to Section 1.5, *Configuration Files* for more details.

13.1.2. Deleting Points

It is not possible to directly delete points from Skkynet VINE. This is because a connecting process may be using that point. Performance does not suffer if there are unused points in the system, but some users prefer to remove them to just keep things tidy. Should a point no longer be in use or requested by any participating program, when VINE is shut down and restarted, the point will no longer appear. For mutliple instances of VINE connected via a tunnel/mirror connection, all of them must be shut down together to ensure that a point is deleted.

13.1.3. Viewing Data Points

You can view the values of all data points with the Data Browser.

13.1.4. Point Size Limits

Skkynet VINE itself does not limit the size of a point data message. The only limits are those imposed by the operating system, and in Windows there are no such limits. If, however, you intend to share data between with Linux or QNX computers, there is a limit of 64000 bytes for QNX and 128000 bytes for

Linux. In any case, bear in mind that very large values will take some time to be transmitted over a network.

13.2. Data Communication Concepts

These basic concepts of data communications will help you understand how Skkynet VINE works.

13.2.1. Send and Receive Data

- **Send/write data:** A program *sends* a value for a data point, and VINE records, or *writes*, the value for that point. This type of communication is synchronous. The send and the write are essentially two parts of a single process, so we use the terms pretty much interchangeably. You can write a value to VINE manually using the Data Browser.

  A typical write command from a program using DDE protocol is **DDEPoke**.

- **Receive/read data:** A program requests to *receive* the value of a data point. VINE then responds by sending the current value of the point. We call this *reading* the value from Skkynet VINE. Again, we sometimes use the two terms interchangeably, and again, this type of communication is synchronous.

  A typical read command from a program using DDE protocol is **DDERequest**.

- **'Automatic' Receive:** It is possible to set up live data channels, where a program receives updates on data points sent from Skkynet VINE. How it works is the program sends an initial request to VINE to register for all changes to a data point. VINE immediately sends the current value of the point, and then again whenever it changes. VINE can receive data automatically in a similar way. This asynchronous type of communication is sometimes referred to as *publish-subscribe*.

  A **DDEAdvise** command sets up this type of connection, which is called an *advise loop*.

13.2.2. Client - Server

Exchanging data with Skkynet VINE is done through a client-server mechanism, where the *client* requests a service, and the *server* provides the service. Depending on the programs it interacts with, VINE is capable of acting as a client, as a server, or as both simultaneously.

The client-server relationship itself does not determine the direction of data flow. For example, a client may read data from the server, or it might write data to the server. The data can flow either way; the client might initiate a read or a write, and the server would respond.

13.2.3. Synchronous and Asynchronous Communication

Every type of communication, natural or man made, comes in two basic forms: *synchronous* or *asynchronous*.

- Synchronous communication means that for each message, the sender expects to get a reply from the receiver, like a telephone call. There is a back-and-forth exchange, so that each party knows that the other is receiving the message. If there is no response, you can be pretty sure that communication didn't occur.

- Asynchronous communication means that a message gets sent but the receiver is not expected to reply, like a radio broadcast or a newspaper.

Each of these communication types has its own value and purpose in data communications, and Skkynet VINE is capable of both. The specific circumstances and application will determine which form of communication you end up using.

13.3. Data Exchange Protocols

Skkynet VINE relays data between programs using OPC, TCP, or DDE. It also tunnels data over a network or the Internet using TCP. This section gives an overview of these protocols.

13.3.1. DDE Protocol

DDE (Dynamic Data Exchange) is a well-established mechanism for exchanging data among processes in MS-Windows. There are three DDE commands for establishing communication with Windows programs such as Excel. Which of these commands you use depends on how you plan to control the flow of data between the spreadsheet and Skkynet VINE.

1. **DDEPoke** writes a data value to Skkynet VINE. For example, to send a value from an Excel spreadsheet to VINE, the **DDEPoke** command is run from within an Excel macro. For more details, please refer to Section 3.2.2, *Method 2 - Writing Excel macros that use the **DDEPoke** command*.
2. **DDERequest** reads a data value from Skkynet VINE. To get that value into Excel, for example, the **DDERequest** command is run from within an Excel macro. For more details, please refer to Section 3.1.2, *Method 2 - Excel Macros using **DDERequest***.
3. **DDEAdvise** creates a connection, called an *advise loop*, that updates a new data value automatically. The advise loop is a unidirectional link, established by a client program that wants to receive data from a server program. The client continues to receive new point values as long as the two programs are running, or until the advise loop is terminated.

   You can use **DDEAdvise** to read data from Skkynet VINE by configuring VINE to act as a DDE Server. For an example using Excel, please refer to Section 3.1.1, *Method 1 - Drag and Drop using **DDEAdvise***.

   Likewise, you can use **DDEAdvise** to write data to Skkynet VINE, by configuring it to act as a DDE Client. For an example using Excel, please refer to Section 3.2.1, *Method 1 - Configuring **DDEAdvise** loops in Skkynet VINE*.

For more information on DDE, please refer to Appendix A, *DDE Overview*.

13.3.2. TCP and Tunnelling/Mirroring

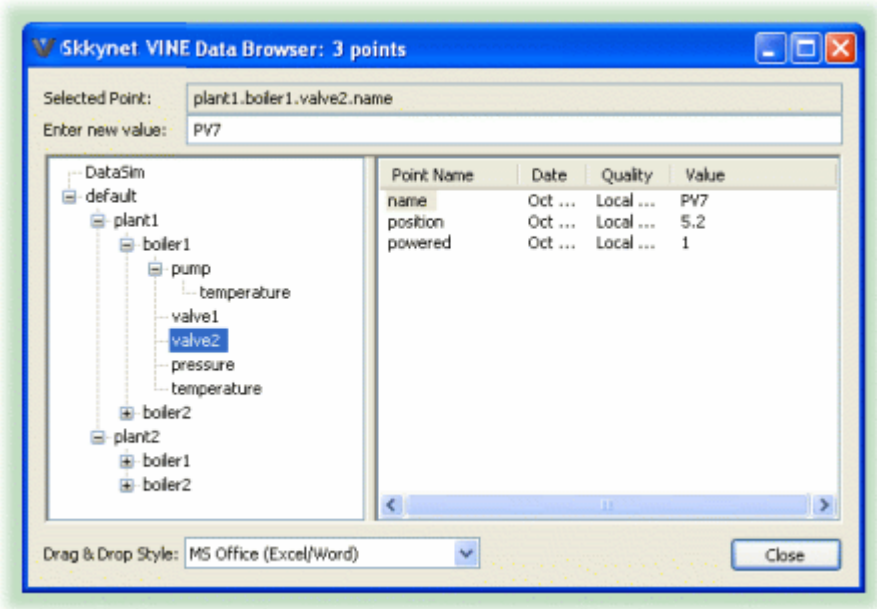
Skkynet VINE uses TCP to communicate over a LAN, WAN, or the Internet. You can join two or more copies of Skkynet VINE together over TCP and share exact copies of the data through data *tunnelling/mirroring*. Tunnelling/Mirroring means that the data and any updates to that data on one VINE are exactly tunneled/mirrored across the network onto any other VINE that is connected. Once a tunnelling/mirroring connection is established each participant maintains and updates an identical data set, as simultaneously as the TCP connection will permit. For more information on tunnelling/mirroring, please refer to Section 15.2, *Tunnel/Mirror*.

13.3.3. The VINE API

The Skkynet VINE APIs for C++, Java, and .NET lets any TCP-enabled program interface with VINE.

13.4. Data Organization

Skkynet VINE offers a hierarchical system to organize your data. The hierarchy is separate from the actual data. It contains points, but is not made up of data, per se. You specify the data model, similar to specifying a class in a programming language, and then you create zero or more instances of that data model. The data hierarchy can be viewed in the left-hand pane of the Data Browser window.



13.4.1. Data Domains

The highest level of the data hierarchy is the data domain. You can create as many data domains as you need, and use them to separate data by user, function, or any other criteria. Points in different data domains can have the same name because each data domain creates a separate namespace. Two data domains you are probably familiar with are `default`, the default data domain, and `DataSim` which holds data from DataSim. All the data domains in VINE are listed in the General option of the Properties window.

The written syntax used by VINE to denote data domains and points is:

```
domain:point
```

In many cases, this is the only level of data organization you will ever need. However, should you desire a more sophisticated way to structure your data, VINE provides a way.

13.4.2. Assemblies, Subassemblies, Attributes, and Properties

Within a data domain, data can be arranged hierarchically as assemblies, subassemblies, attributes, and properties. Each assembly can have zero or more attributes and zero or more subassemblies, and each attribute can have zero or more properties. Subassemblies can have subassemblies. You can think of assemblies and subassemblies as branches in a tree, and attributes as the leaves. Here is an example of what a tree might look like:

```
Data Domain
    Assembly
        Subassembly (zero or more)
            Attribute  (zero or more)
                Property  (zero or more)
        Attribute...
        Attribute...
        Attribute...
            Property...
            Property...
            Property...
        Subassembly
            Subassembly
                Attribute...
                  Property...
                  Property...
            Attribute...
            Attribute...
                Property...
    Assembly...
```

and so on.

The written syntax for all of these levels uses a dot (`.`) to divide the names, rather than a colon that was used for the data domain name. Hence, the syntax of point in a property in an attribute in a subassembly in an assembly in a data domain would be:

`domain``:`*`assembly.subassembly.attribute.property`*

Properties describe the attributes in more detail. An attribute can have a *default property* such that if you interact with the attribute point directly you will in fact be interacting with its default property. For example, an item might be `plant.temperature`, with properties `value`, `highlimit`, `units`. This would create 4 tags:

```
plant.temperature
plant.temperature.highlimit
plant.temperature.units
plant.temperature.value
```

The tags:

```
plant.temperature
plant.temperature.value
```

are aliases of one another. Both refer to the default property of `plant.temperature`. If you specify no property at all for an item, the item takes on the default property.

13.4.3. Attributes and Types

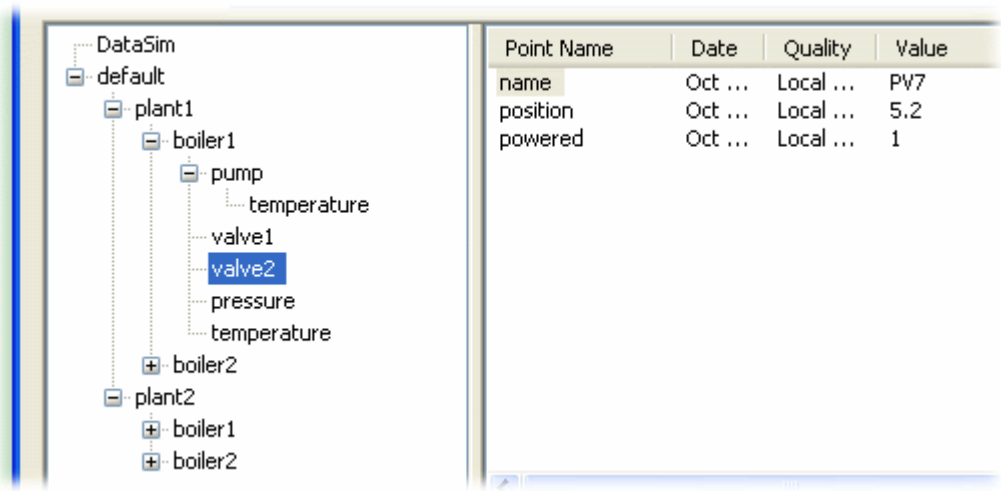
It is common for attributes to contain the same type of information. For example, all temperatures in a system are likely to share units, high alarm level, and value. To avoid repeating this information for each and every temperature in the system, we use a *type*. A type is the prototype, or class, of an attribute. You define a type and its properties first, and then define attributes of that type on assemblies. When the assembly is instantiated, its attributes are instantiated by creating an attribute and then assigning the properties to it that are associated with the attribute's type.

There is an alternative to using types and attributes as described here, a *private attribute*. A private attribute provides a one-command (**private_attribute**) means of creating an attribute on an assembly without having to define a type. This prevents the attribute properties from being shared across more than one attribute in the assembly or in other assemblies, but is easier to use when defining simple hierarchies. (See Example 2.)

13.4.4. Example 1: Attributes and Types

Suppose we want to create a hierarchical data model like this: We have a control system consisting of process areas, that we will call "Plants". Each plant contains 2 boilers, and each boiler has a pump and 2 valves. Each boiler measures temperature, pressure and level. Each pump measures speed, on/off state and operating temperature. There are two types of valves - a normal one that only measures position, and another that also verifies that it has power applied to it. Temperatures have a value and a high alarm limit.

The sample file `plant.cfg` shown below included in the VINE distribution will create a point hierarchy in the `default` data domain that looks like this:



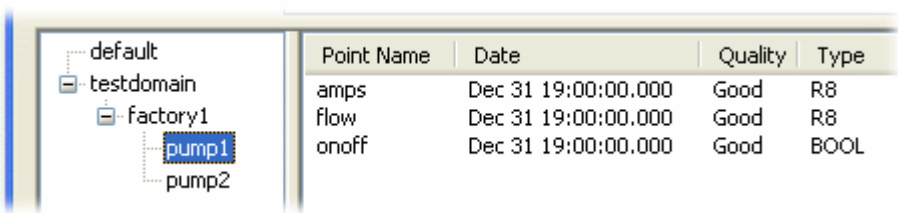
You can have VINE load this `plant.cfg` configuration file on startup (see Section 1.5, *Configuration Files*).

```
;;; Create a generic object to share all of the common properties and
;;; attributes in the model.  Give it a common property, called "name"

(assembly default Object)
(property default Object AUTO name string rw "unnamed" 100)

;;; Create a temperature attribute that can be shared by boilers and pumps.
;;; It has three properties: value,highlimit,units

(type default Temperature)
(property default Temperature AUTO value R8 rw 0 100)
(property default Temperature AUTO highlimit R8 rw 120 100)
(property default Temperature AUTO units STRING rw "C" 100)
(defaultprop default Temperature value)

;;; Create a pressure attribute for boilers.

(type default Pressure)
(property default Pressure AUTO value R8 rw 0 100)
(property default Pressure AUTO units STRING rw "kPa" 100)
(defaultprop default Pressure value)

;;; Create a plant model, sharing the properties and attributes of "object"

(assembly default Plant Object)

;;; Create a boiler model, as an "object"

(assembly default Boiler Object)
(attribute default Boiler temperature Temperature)
(attribute default Boiler pressure Pressure)

;;; Create a pump model, as an "object"
```

```
(assembly default Pump Object)
(attribute default Pump temperature Temperature)
(property default Pump AUTO speed R8 rw 0 100)
(property default Pump AUTO state I4 rw 0 100)

;;; Create a valve object.  It has a property, position, directly
;;; attached to the assembly.  We do not need an attribute unless
;;; there is more than one property to be associated with it.  In
;;; this example position has only a value, without limits or
;;; units.

(assembly default Valve Object)
(property default Valve AUTO position R8 rw 0 100)

;;; Create a specialization of a Valve that also measures whether
;;; the valve is powered.

(assembly default Powervalve Valve)
(property default Powervalve AUTO powered I4 rw 0 100)

;;; Create the hierarchy in the model

;;; Plants have two boilers, named boiler1 and boiler2
(subassembly default Plant Boiler boiler1)
(subassembly default Plant Boiler boiler2)

;;; Boilers have one Pump, named pump
(subassembly default Boiler Pump pump)

;;; Boilers have a normal valve and a powered valve, named valve1
;;; and valve2 respectively.
(subassembly default Boiler Valve valve1)
(subassembly default Boiler Powervalve valve2)

;;; Create two plants named plant1 and plant2.  These actually
;;; create the data points in the DataHub and arrange them in
;;; the hierarchy specified above.  Up to this point, the commands
;;; have just been building a model.  These calls instantiate the
;;; model.

(instance default plant1 Plant)
(instance default plant2 Plant)
```

13.4.5. Example 2: Private Attributes

Here is a simpler example for creating a hierarchical data model using private attributes. This models a control system in a factory with 2 pumps. The sample file shown below will create a point hierarchy in the `default` data domain that looks like this:




```
; Create two assemblies.
(assembly testdomain factory)
(assembly testdomain pump)

; Create two subassemblies.
(subassembly testdomain factory pump pump1)
(subassembly testdomain factory pump pump2)
```

```
; Assign private attributes.
(private_attribute testdomain pump flow R8 rw 0 100)
(private_attribute testdomain pump amps R8 rw 0 100)
(private_attribute testdomain pump onoff BOOL rw 0 100)

; Instantiate the model.
(instance testdomain factory1 factory)
```

Chapter 14. Optimizing Data Throughput

Skkynet VINE has a wide range of configuration options. Among these there are several settings that will optimize data throughput. These are explained in this chapter.

14.1. Binary Mode Tunnel/Mirror (TCP) Connections

TCP/IP connections to Skkynet VINE can be either ASCII or binary mode. A large part of the CPU cost of transmission is marshalling messages (constructing messages at the source and parsing them at the destination). The binary mode is more efficient in both network bandwidth and CPU usage for both the sender and the receiver. Binary mode requires that the CPU architecture of the sender and the receiver agree, so you can only use this mode if you are running both the sender and the receiver on an Intel x86 CPU. The CPU gain could be as much as 50% when using binary mode. Numeric data benefits most from this option.

How to Optimize

- For tunnelling connections, always use binary mode if possible. Please refer to How to Optimize, Procedure 14.2, *Binary mode transmission* for details.
- For TCP/IP connections using the C++ API, always use binary mode if possible. Please refer to How to Optimize, Section 14.7.2, *DataHub C++ API* for details.
- Always use binary mode when writing programs in Linux or QNX with the Cogent C API. Please refer to Optimizing Throughput in the Cogent C API manual for details.

14.2. Tunnel/Mirror (TCP) Connections for Slow Networks

Skkynet VINE uses a heartbeat to determine the status of the network connection. The tunnel/mirror slave sends a special heartbeat message to the master at specified time intervals, to detect network failures. If the master does not respond within a certain timeout period, the slave changes the status of its connection to `Disconnected` and attempts to reconnect.

The timout value is usually at least twice the heartbeat rate, and the default setting is five times the heartbeat rate. However, if you have a very slow or irregular network connection, the best thing to ensure that the connection remains open is to override the VINE timeout, and use the TCP timeout mechanism. It takes longer to detect a network failure, because it waits as long as the TCP stack waits. It means the slowest recognition of a broken link, but doesn't give up until TCP says the link is really dead.

How to Optimize

- To ensure the longest-lasting TCP connection over a very slow network, you need to set the heartbeat Timeout to `0`. This will cause VINE to ignore its heartbeat settings and rely on the TCP implementation for detecting a broken connection. Please refer to How to Optimize, Procedure 14.3, *Slow network* for details.

14.3. Old Value Queuing

Skkynet VINE maintains a queue of old values for all registered points for each client. The depth of this queue is variable. The purpose of queuing old values is to reduce the chance that a data change will be missed during bursts of abnormally high data flow.

For example, if a switch is turned on, then off, then on again very rapidly, the data might arrive at VINE so quickly that it has no opportunity to send it where it needs to go before the next value arrives. If this happens, VINE may only transmit the final "on" value and the client will not notice that there was in fact an on-off-on transition.

Skkynet VINE can maintain a short queue to reduce the probability of this happening. If the queue is at least three values deep, VINE will send the on-off-on transition even if it knows that the first two values are already stale.

Old value queuing is harmless so long as periods of abnormally high data flow are short. If the data flow rate is high enough that VINE can never keep up, the effect is that the old value queue will always be full, no matter how long or short it is. The CPU cost of maintaining even a short queue in a sustained overload situation is very high, and depends on the queue depth. See also Section 14.6, *CPU Saturation*.

How to Optimize

- If your system runs at CPU saturation, eliminate the old value queue if at all possible for TCP/IP connections. Please refer to How to Optimize, Procedure 14.4, *Old value queueing and un-buffered delivery* for details.
- If your system runs at CPU saturation, eliminate the old value queue if at all possible for the Gamma scripting engine. Please refer to How to Optimize, Section 14.7.3, *Gamma scripts* for details.

14.4. Un-Buffered Delivery

Skkynet VINE buffers data that will be transmitted to a client such that if it knows that more incoming data is available, it will hold off outgoing transmissions until it has a complete data set to send onward to the client. This does not introduce extra latency because VINE will only accumulate data destined for a client that arrives together in an incoming message.

This buffering greatly increases efficiency by reducing thread context switching and by giving its protocol-specific data transmitters an opportunity to collect more than one data change into a single outgoing message.

One of the side-effects of buffering is that the old value queue will be more likely to fill. If your application is very sensitive to every change of value, then it may be necessary to turn off the buffering. The CPU penalty for turning off buffering is very high, perhaps as much as 200% in heavy load conditions.

How to Optimize

Do not use un-buffered data delivery in high load conditions unless you absolutely must. Please refer to How to Optimize, Procedure 14.4, *Old value queueing and un-buffered delivery* for details.

14.5. Screen Output

Output to the screen can use a huge amount of processing time. The worst offenders are multi-line text boxes where text is constantly being added and scrolled. If you think that the CPU usage for Skkynet VINE is too high, close all Event Log and Script Log windows.

You may also find that the Data Viewer window uses too much CPU if the data is changing very rapidly or if the number of data points visible in the right-hand pane of the Data Viewer is very large. Try closing the Data Viewer to reduce CPU load. If you need to monitor some data in the Data Viewer, consider using bridging to collect the subset of data points that you need into separate data domain. Viewing that subset of the data will consume less CPU and may have the added benefit of being more convenient.

How to Optimize

- Close the Event Log windows when not in use.
- Close the Script Log window when not in use.
- Close the Data Viewer when not in use, or use bridging to create a smaller subset of data in a separate data domain.

14.6. CPU Saturation

When your system is running with maximum CPU utilization, the transmitting and receiving threads within Skkynet VINE must all share the CPU available. This will cause data to be queued more frequently, and will cause VINE to take measures to cope with the lack of CPU. VINE treats a high-CPU condition as if the various connections cannot consume data as quickly as it is available. It will begin tracking and ultimately discarding old data values, and will more aggressively accumulate data changes into larger messages wherever possible. VINE's goal is to ensure that latency remains low, that all clients continue to receive data, and that the clients always receive the most recent data that is available.

This effort to cope with reduced resource availability also uses more CPU, somewhat further increasing the system load. If old value queues are deep, the system can reach a "tipping point" from which it is difficult to recover without severely reducing the input data rate.

How to Optimize

Avoid running your system at maximum CPU capacity.

14.7. How to Optimize

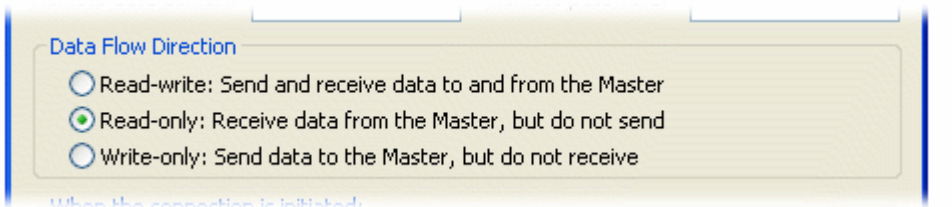
14.7.1. Tunnel/Mirror (TCP) connections

Read-only connections

On the **slave** side of the connection, you can determine the data flow direction for the connection. If the data flow is to be one-way, from the master to the slave (i.e. the slave will only read from the master, not write), you will get the fastest performance by configuring the connection "read-only", as follows:

1. On the VINE that is making the *slave* side connection, right click on the VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.
3. In the Data Flow Direction section, select Read-only: Receive data from the Master, but do not send.


Data Flow Direction
Read-write: Send and receive data to and from the Master
Read-only: Receive data from the Master, but do not send
Write-only: Send data to the Master, but do not receive

4. Click Apply.

Binary mode transmission

Ensure that the **slave** side of the tunnel/mirror connection is set to use binary mode transmission:

1. On the VINE that is making the *slave* side connection, right click on the VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.
3. In the Tunnel/Mirror Slave section, highlight the host name of the tunnelling master, and click the Edit button to open the Tunnel/Mirror Master Configuration window.
4. Check the Transmit point changes in binary box.

Connection Properties
Replace incoming time stamp with the local current time
Transmit point changes in binary (faster, x86 CPU only)
Target is a Cogent Embedded Toolkit server
Heartbeat (ms):
0
Retry Delay (ms):
5000

5. Click Apply.

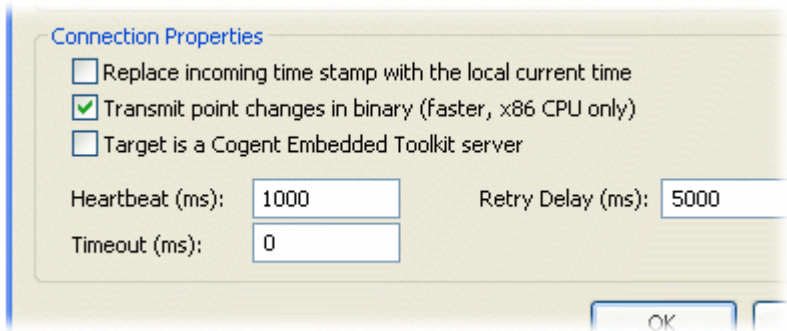
For more information, please refer to Section 14.1, *Binary Mode Tunnel/Mirror (TCP) Connections*.

Slow network

To optimize performance for a slow network, you can set the heartbeat timeout to `0` on the **slave** side, as follows:

1. On the VINE that is making the *slave* side connection, right click on the VINE system-tray icon and choose Properties.
2. In the Properties window, select Tunnel/Mirror

.
3. In the Connection Properties section, enter `0` in the Timeout entry field.



4. Click Apply.

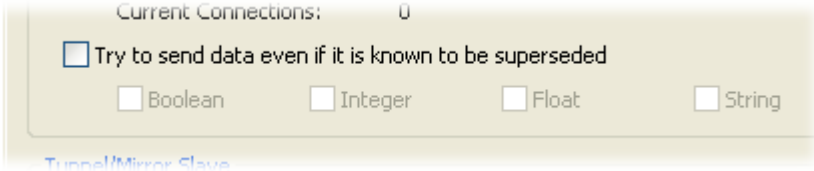
For more information, please refer to Section 14.2, *Tunnel/Mirror (TCP) Connections for Slow Networks*.

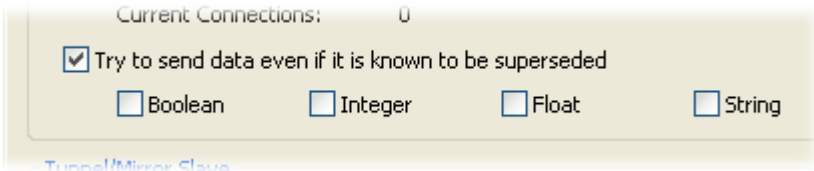
Old value queueing and un-buffered delivery

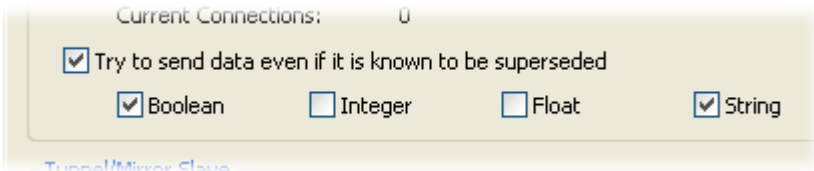
Configure the **master** side of the tunnel/mirror connection:

1. On the VINE that is the *master* for the tunnel/mirror connection, right click on the VINE system-tray icon and choose Properties.

2. In the Properties window, select Tunnel/Mirror

.
3. In the Tunnel/Mirror Master section, you can choose between one of three states:
   - **No queuing with buffered delivery.** This is the fastest state. To do this, un-check the option Try to send data even if it is known to be superseded in the Tunnel/Mirror configuration tab:


   - **Queuing with buffered delivery.** This is a reasonable compromise that will keep up to three old values for each data point if there is a short burst of heavy data traffic. If there are more than three values for a point outstanding, the oldest will be discarded. To do this, check the option Try to send data even if it is known to be superseded, but do not select any of the data types below it:


   - **Queuing with un-buffered delivery.** This is the slowest mode, but also the one least likely to discard old values during short periods of heavy data traffic. In this mode there is a queue of up to 3 old values for each data point. In addition, you may choose to force the data transmission when a point with any of the specified types changes. The goal here is to possibly allow buffering (and hence, possible discarding of old values) for some types, while attempting to preserve all changes for other types. For example, here we have chosen to force an outbound transmission if any boolean or string value changes, but to buffer any changes in floating point and integer types:


4. Click Apply.

For more information, please refer to Section 14.3, *Old Value Queuing* and Section 14.4, *Un-Buffered Delivery*.

14.7.2. DataHub C++ API

Binary mode connections

1. In your program, call the method

   ```
   CDataHubConnector::sendBinaryPointMessage(bool enable)
   ```

   to enable or disable binary messages.

For more information, please refer to Section 14.1, *Binary Mode Tunnel/Mirror (TCP) Connections*.

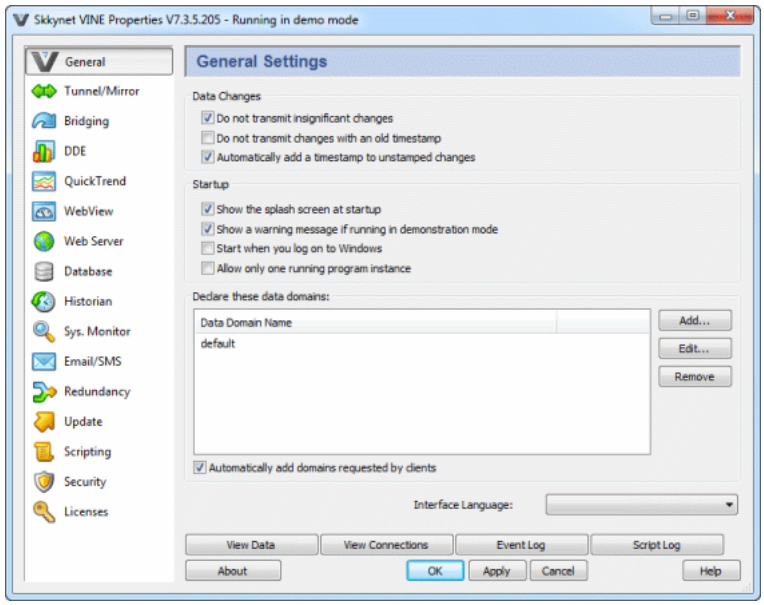
14.7.3. Gamma scripts

The Gamma engine services all VINE scripts through a single queue, for the sake of efficiency. This means that any changes you make to the default behaviour will apply to all running scripts. By default, the Gamma engine runs with a 3-deep queue and buffered transmission. This is equivalent to the queuing with buffered delivery option (above) for TCP/IP connections. As of Skkynet VINE version 6.4.2, you can modify the behaviour of the queuing and the buffering via Gamma function calls:

- **`set_point_queue_depth`** lets you specify the depth of the per-point queue. It is wise to keep this value small. Please see set_point_queue_depth in the VINE Scripting manual for details about this function.
- **`get_point_queue_depth`** determines the current point queue depth for Gamma. Please see get_point_queue_depth in the VINE Scripting manual for details about this function.
- **`set_point_flush_flags`** sets which data types will cause the point buffer to immediately be transmitted to the Gamma engine. Please see set_point_flush_flags in the VINE Scripting manual for details about this function.

Chapter 15. Properties Window

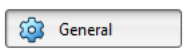
This is where you can configure Skkynet VINE.



For all the options (General, Data Domains, DDE, etc.) in this window:

- The OK button applies changes and closes the window.
- The Apply button applies changes but leaves the window open.
- The Cancel button closes the window without applying any changes.
- The Help button opens the help window for the current option.

15.1. General



This first option in the Properties Window lets you control how changes to data are transmitted from Skkynet VINE.

Data Changes

Do not transmit insignificant changes will reduce traffic by allowing only significant changes to the data to be sent. A change is *significant* if a property of the point other than the time-stamp changes. That normally means a change to either the value or the quality of the point. A change in only the time-stamp is considered insignificant. Some polled data sources change the time-stamp on each cycle, even if the value doesn't change. If network bandwidth is a concern, you can use this option to update the point only when the value has changed.

Do not transmit changes with an old timestamp allows only current or future changes to be sent.

Automatically add a timestamp to unstamped changes stamps the current time onto any changes that haven't already been time stamped.

This should stay checked unless you have specific reason to uncheck it. Unchecking it may cause changes made through DataSim, the Data Browser, and other programs to receive timestamps of 0 (`Dec 31 19:00:00:00.000`). If this button is *unchecked* and the Do not transmit change with an old timestamp is *checked*, then any changes with a 0 timestamp won't get transmitted at all.

Startup

Show the splash screen at startup lets you hide or show the startup screen with the VINE image.

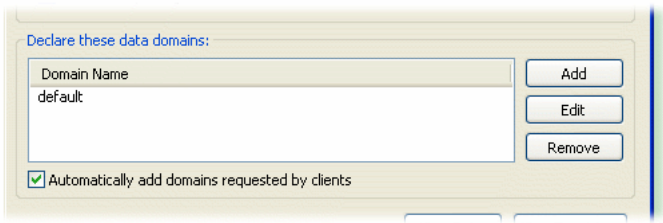
Show a warning message if running in demonstration mode lets you hide or show the message telling you the demo will terminate in one hour.

Start when you log on to Windows causes VINE to start up whenever you log on to Windows.

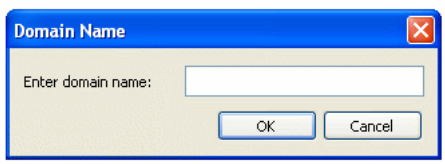
Allow only one running program instance prevents more than one Skkynet VINE from running at the same time.

Declare these data domains

In this area you can add, edit, or remove data domains for Skkynet VINE. For more information about data domains, please refer to Section 13.4.1, *Data Domains*.



To add a data domain, click the Add button and fill in the name in the Data Domain Name Window:



To change a data domain, double-click it or select it and click the Edit button. To remove a data domain, highlight it and click the Remove button.

Checking the Automatically add data domains requested by clients box automatically adds a data domain whenever a client requests it. If for some reason you want to limit the data domains to those listed, you should make sure this box is not checked.

Other Options

The Interface Language list lets you choose a language for the interface. If you don't find your language, you can contact Cogent for instructions on how to add a translation to the VINE source code.

The View Data button starts Data Browser.

The Event Log button starts Event Log.

The Script Log button starts Script Log.

15.2. Tunnel/Mirror

The Tunnel/Mirror option lets you configure Skkynet VINE to act as a master or slave for *tunnelling/mirroring*. Tunnelling/Mirroring allows you to send OPC or DDE data across a network robustly and securely. Tunnelling is done over TCP, which provides connectivity across a network or over the Internet.

Tunnelling and Mirroring

The Skkynet VINE tunnelling connection is sometimes referred to as a *mirroring* connection. Mirroring means that the data and any updates to that data on one VINE are exactly mirrored across the network onto the other VINE, and vice-versa. For all practical purposes, tunnelling and mirroring are identical. People working with OPC tend to use the term "tunnelling" while people from other backgrounds often say "mirroring". So Cogent uses "tunnelling" for Skkynet VINE, and "mirroring" for other Cogent products. For example, Skkynet VINE can tunnel (mirror) data to the Cascade DataHub running in the Linux or QNX operating systems. Please refer to the Mirroring Data section of the Cascade DataHub for Linux and QNX manual for details about those operating systems.

Direct TCP connections

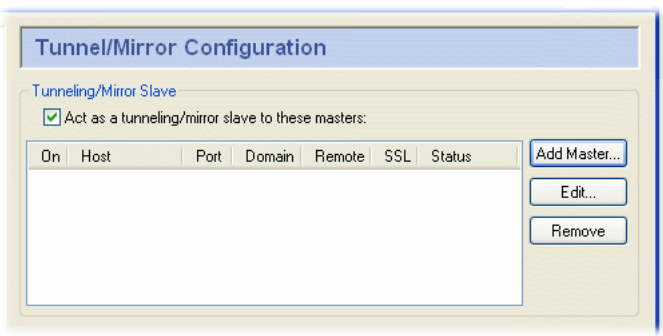
In addition to tunnelling, Skkynet VINE can accept direct connections from any TCP client using the Skkynet VINE APIs for C++, Java, and .NET, such as DataSim or other, custom applications.

Master and Slave

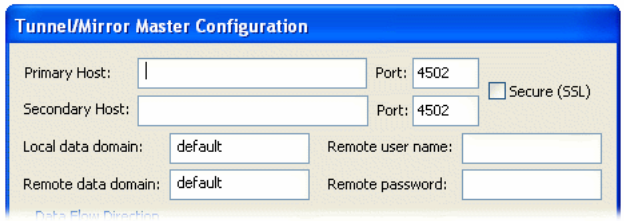
We identify the two tunnelling/mirroring VINEs as *master* and *slave*. The only difference between the master VINE and slave VINE is that the slave initiates the connection. Once the connection is established, they function exactly the same. It is possible for a VINE to be both tunnelling/mirroring slave and master simultaneously—acting as a slave to one or more VINEs and a master to one or more others. For slave mode you need to specify each master.

Tunnel/Mirror Slave

Check the Act as a tunnelling/mirror slave to these masters box to have Skkynet VINE act as a slave.



To add a master for this mode, click the Add Master... button. To edit a master, double-click it, or select it and press the Edit... button. Either button opens the Tunnel/Mirror Master window:



Type in the following information:

Primary/Secondary Host

The name or IP address of the host computer. This slave VINE will alternate attempts to connect first on the primary host, then on the secondary host, back and forth until a connection is made. The secondary host is optional, and if not entered, all attempts to reconnect will be on the primary host. If the connection is interrupted, VINE will again alternate attempts at reconnection on the primary and secondary hosts.

This feature is not recommended for redundancy because it only checks for a TCP disconnect. The VINE Redundancy feature, on the other hand, provides full-time TCP connections to both data sources, for instantaneous switchover when one source fails for any reason. There is no need to start up the OPC server and wait for it to configure its data set. You can also specify a preferred source, and automatically switch back to that data source whenever it becomes available. By contrast, the primary and secondary host in the tunnel can act as a primitive form of redundancy, but will only switch on a connection failure at the TCP level, which is only one sort of failure that a real redundancy pair must consider.

Port

The port number or service name as entered in the Master service/port entry box of the master on the remote computer.

Secure (SSL)

You can establish a secure connection using SSL tunnelling as long as the tunnelling master VINE you are attempting to connect to has been configured for secure connections. (See above.)

Local data domain

The local Skkynet VINE data domain for this slave. It is common, but not necessary, to create or use an existing local data domain that has the same name as the remote data domain.

Remote data domain

The name of the remote Cascade DataHub data domain, which is the tunnelling master. Point names will be mapped from that data domain into the local data domain, and vice versa.

Remote user name

The user name for TCP security, established on the tunnelling master, using the VINE Security option in the Properties window.

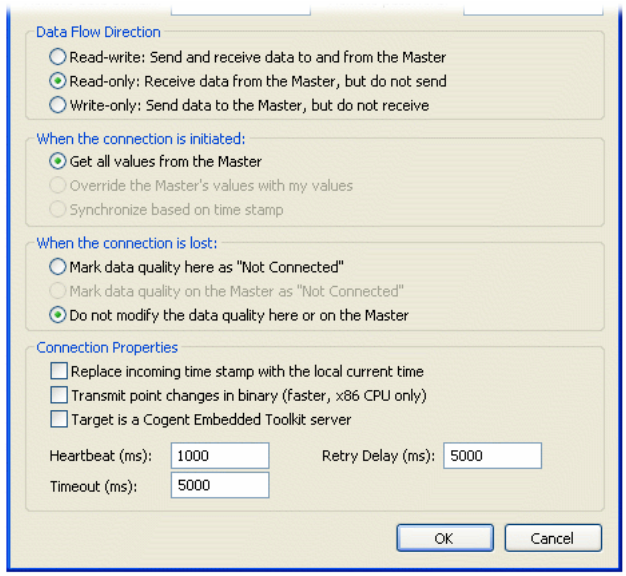
Remote passwork

The password for TCP security, established on the tunnelling master, using the VINE Security option in the Properties window.

There is a VINE running on Cogent's server that you can connect to for testing. Here are the parameters you will need to enter for it:

- **Primary Host:** `developers.cogentrts.com`
- Port: `4502`
- **Local data domain:** `test`
- **Remote data domain:** `test`

You now have several options for the mirrored connection.


Data Flow Direction
Read-write: Send and receive data to and from the Master
Read-only: Receive data from the Master, but do not send
Write-only: Send data to the Master, but do not receive
When the connection is initiated:
Get all values from the Master
Override the Master's values with my values
Synchronize based on time stamp
When the connection is lost:
Mark data quality here as "Not Connected"
Mark data quality on the Master as "Not Connected"
Do not modify the data quality here or on the Master
Connection Properties
Replace incoming time stamp with the local current time
Transmit point changes in binary (faster, x86 CPU only)
Target is a Cogent Embedded Toolkit server
Heartbeat (ms): 1000
Retry Delay (ms): 5000
Timeout (ms): 5000
OK
Cancel

1. **Data Flow Direction** lets you determine which way the data flows. The default is read-only data flow from master to slave, but you can set up a read-write or write-only connection by choosing those options.

   To optimize throughput, check the Read-only: Receive data from the Master, but do not send option. Only do this if you actually want a read-only connection. If you do not require read-write access, a read-only tunnel will be faster.

2. **When the connection is initiated** determines how the values from the points are assigned when the slave first connects to the master. There three possibilities: the slave gets all values from the master (the default), the slave sends all its values to the master, or the data from master and slave gets synchronized. The availability of these options depends on the data flow direction selected above.

3. **When the connection is lost** determines where to display the data quality as "Not Connected", on the master, on the slave, or neither.

   If you have configured When the connection is initiated as Synchronize based on time stamp (see above), then this option must be set to Do not modify the data quality here or on the Master to get correct data synchronization.

4. **Connection Properties** gives you these options:

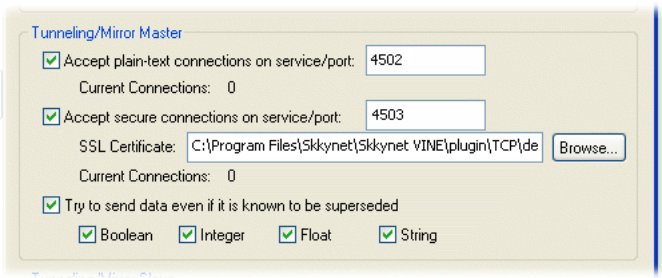
   - **Replace incoming timestamp...** lets you use local time on timestamps. This is useful if the source of the data either does not generate time stamps, or you do not trust the clock on the data source.
   - **Transmit point changes in binary** gives users of x86 CPUs a way to speed up the data transfer rate. Selecting this option can improve maximum throughput by up to 50%, depending on the type of data being transmitted. This option uses a more efficient message encoding scheme than the default ASCII encoding, but it will only work if both sides of the tunnel are running on an x86

architecture CPU. This would be typical of Windows communicating with Linux or QNX, or with another Windows computer. Numeric data benefits most from this option.

- **Target is a Cogent Embedded Toolkit server** allows this slave to connect to an Embedded Toolkit server rather than to another VINE.
- **Heartbeat** sends a heartbeat message to the master every number of milliseconds specified here, to verify that the connection is up. Setting this value to `0` stops the heartbeat from being transmitted.
- **Timeout** specifies the timeout period for the heartbeat. If the slave VINE doesn't receive a response from the master within this timeout, it drops the connection. You must set the timeout time to at least twice the heartbeat time. Setting this value to `0` will cause VINE to rely on the TCP implementation for detecting a broken connetion. This can be useful when your network connection is very slow. Please refer to Section 14.2, *Tunnel/Mirror (TCP) Connections for Slow Networks* for details.
- **Retry** specifies a number of milliseconds to wait before attempting to reconnect a broken connection.

Tunnel/Mirror Master

You can configure your VINE to act as a master for either plain-text tunnelling, secure tunnelling using SSL, or both. Each mode uses a separate port number or service name.



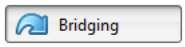
If you enter a name for the service/port instead of a number, that name must be listed in the Windows `services` file. Please refer to The Windows Services file Appendix for details.

VINE installs an SSL Certificate for you. If you wish to move it or use a different one, you can change the directory path here. The SSL implementation uses the default SSL-3 encryption cipher: DHE-RSA-AES256-SHA. This is a 256-bit encryption. The server and client negotiate the best encryption based on what both can support. VINE does not validate the SSL certificate with any outside certificate authority. It uses the SSL connection for encryption only, not authentication.

You can also configure the master to attempt to send "old" data (superseded by more recent data). Check any or all of Boolean, Integer, Float, or String that apply to the kind of superseded data that you wish to have sent.

To optimize throughput using this option, please refer to Section 14.7, *How to Optimize*.

15.3. Bridging

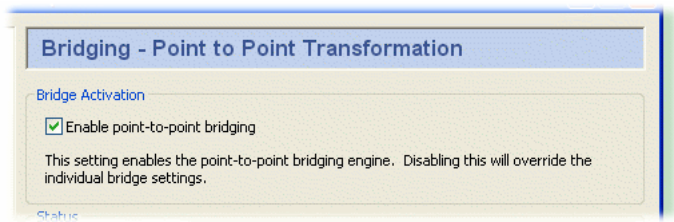


The Bridging option lets you configure Skkynet VINE to configure data bridging. Bridging means connecting points from two different VINE clients so that when one point changes, its value gets written to the bridged point.

You can configure the bridge to be one-way in either direction, or bidirectional. In addition, you can transform the data as it passes through VINE, using linear transformations.

Bridge Activation

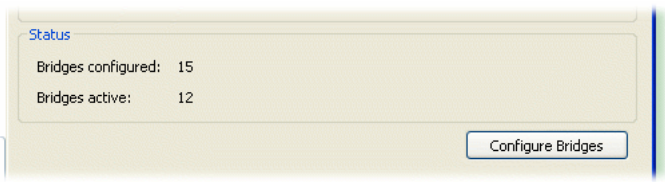
This setting enables and disables bridging globally for the whole Skkynet VINE program.


Bridging - Point to Point Transformation
Bridge Activation
Enable point-to-point bridging
This setting enables the point-to-point bridging engine. Disabling this will override the individual bridge settings.

Ensure the box is checked to enable bridging. Uncheck the box to disable bridging.

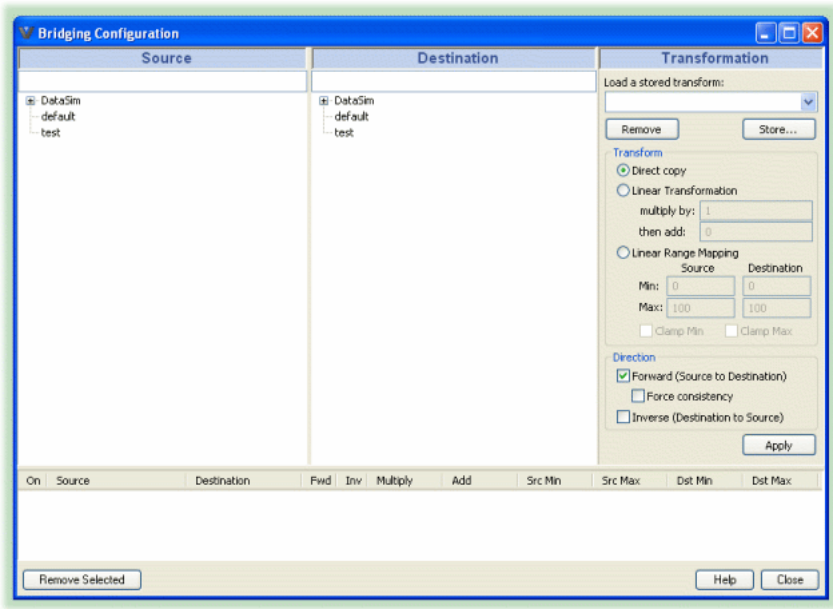
Status

This shows the total number of bridges that have been configured, and how many of them are currently active.



Configure Bridges

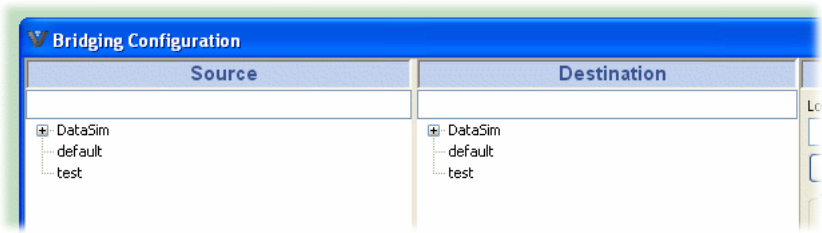
Click the Configure Bridges button to open the Bridge Configuration window:



The following sections give an overview of bridge configuration. For step-by-step instructions, please refer to Section 5.2, *Configuring Bridges*.

Point Selection

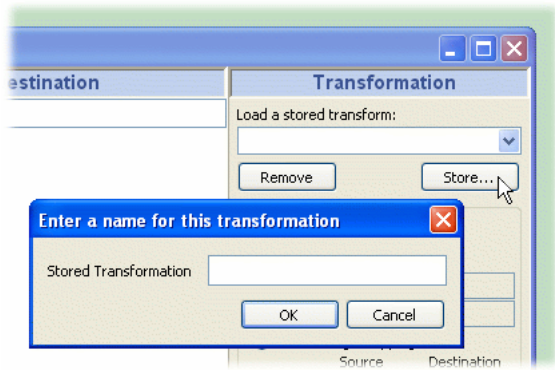
This is where you select the points to be bridged—a source point and a destination point.



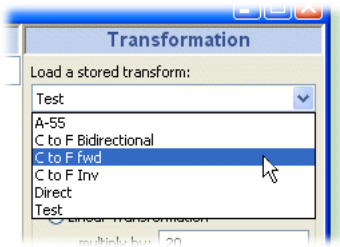
You can click on the point you need, or enter the name in the data-entry box at the top of the column.

Storing Transformations

You can store transformations and retrieve them by name later on.



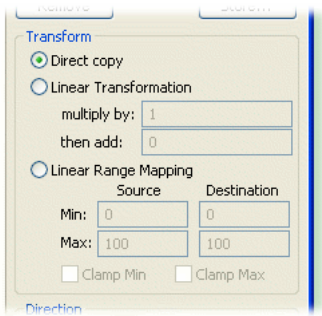
To save a transformation, click the Store... button and enter a name in the pop-up box. Once stored, the transformation will become available by name in the drop-down list.



To load a transformation, simply select its name from the list.

Transform

Specify the type of transformation by clicking Direct copy, Linear Transformation, or Linear Transformation.



- Direct copy makes no transformations. It just copies the point.
- Linear Transformation lets you multiply by one value and add another value, such as in the equation `y = mx + b` where the destination point is `y`, the source point is `x`, the multiply by value is `m`, and the then add value is `b`. For example to transform a Celsius source point to a Fahrenheit destination point, you would multiply by `1.8` and add `32`, or

  ```
  Fahrenheit = (1.8 X Celsius) + 32
  ```

  If you have selected the Inverse direction for a transformation, you will get the inverse of the transformation. In this example, you would get a conversion from Fahrenheit to Celsius, or the results of this equation:

  ```
  Celsius = (Fahrenheit - 32) / 1.8
  ```

  As an alternative to entering transformation values, VINE also offers Linear Range Mapping.

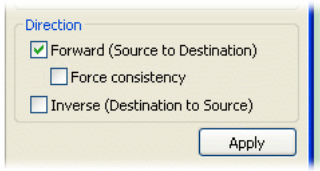
- Linear Range Mapping lets you enter a range for the source and destination, and VINE automatically calculates the corresponding linear transformation. For example, to create the same Fahrenheit to Celsius transformation, you could use the defaults of `0` and `100` for the Min and Max of the source point. Then you would enter `32` and `212` for the Min and Max of the destination point. As soon as you make these entries, the correct values get entered automatically in the Linear Transformation.

  When you use linear range mapping, you can limit the transformed value to the maximum and minimum by checking the Clamp boxes. The clamps get applied to the point being changed, ie. to the

destination point for forward direction, to the source point for inverse direction, and to both points for bidirectional bridges.

Direction

Decide which direction you want the bridge to apply.



- Select Forward to change the destination point when the source point changes, but not change the source when the destination changes. If you select Force consistency with this option, and if the destination point gets changed for some reason, then VINE will attempt to force its value to be consistent with the source point value.
- Select Inverse to change the source point if the destination point changes, but not vice-versa.

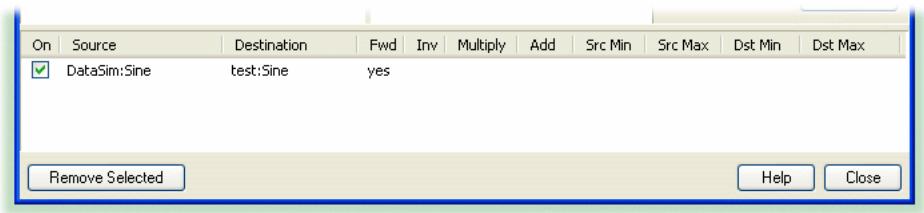
Selecting Inverse will apply the inverse of the transformation, as explained above.

- Select *both* Forward and Inverse for a bidirectional bridge, where either point changes whenever the other point changes. This combination will deselect Force consistency to eliminate the possibility of conflicting behavior.

Click the Apply button to create and activate the bridge. VINE will create the bridge and update the bridged points immediately.

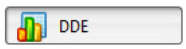
Point Display

Here you can see all the bridges that exist in the system, and the significant information about them.



If you click on a transformation, the source point, destination point, and transform information get displayed in their respective panels. Use the check box at the front of each bridge to activate or deactivate that particular bridge.

15.4. DDE

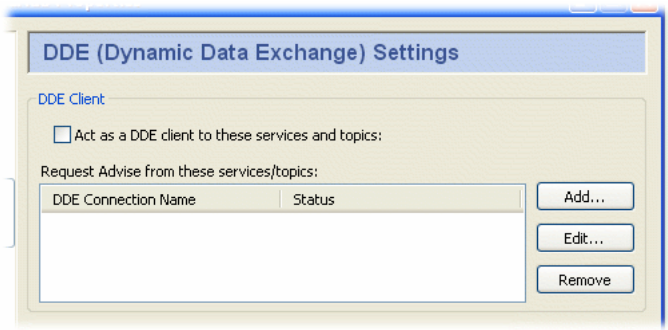


The DDE option lets you configure Skkynet VINE to act as a DDE client and/or DDE server for **DDEAdvise** messages. For more information on DDE, please refer to Section 13.3.1, *DDE Protocol* and Appendix A, *DDE Overview*.

DDE Client

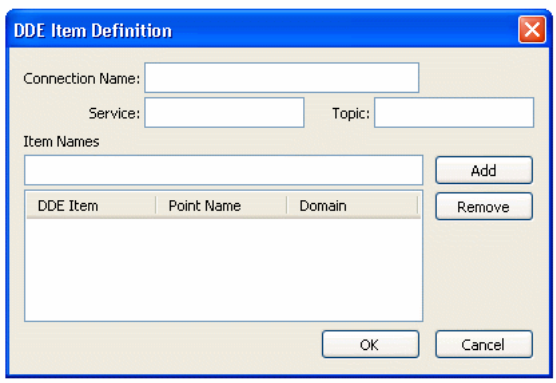
To act as a DDE client, check the Act as a DDE client box.

For best performance, ensure that a DDE server is running when using VINE as a DDE client. A DDE client can consume substantial system resources trying to connect if a DDE server is not available.



To add a new service or topic, press the Add button. To edit a service/topic, double-click it or select it and press the Edit. To remove a service/topic, select it and press the Remove button.

Double-click a selection, or pressing the Add or Edit button opens the DDE Item Definition Window:



Here is what must be entered:

Connection Name

A name used by Skkynet VINE to identify the connection. There should be no spaces in the name. It doesn't matter what name is chosen, but it should be unique to other connection names. Each connection can have only one Service and one Topic, but it can have multiple Items.

Service

The service name of the DDE server to this client.

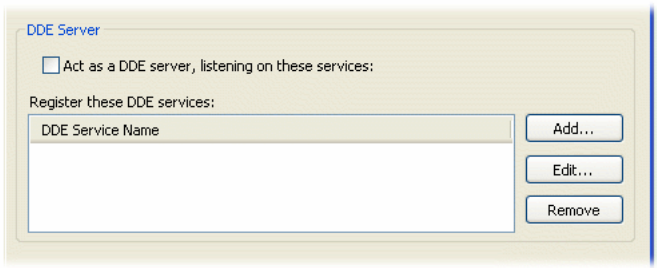
Topic

The DDE topic under which the data will be sent.

Item Names

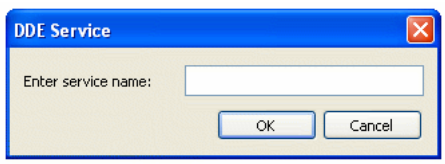
The DDE item name, which corresponds to a point name in Skkynet VINE. If that point doesn't exist in VINE, it will be created. Enter a name for each item and press the Add button to add items. To remove an item, highlight it and press the Remove button. Once all the information is entered correctly, press the OK button to enter the definition.

DDE Server



To have Skkynet VINE act as a DDE server, check the Act as a DDE server box. The default service name is `vine`.

To add a DDE service name to the list, click the Add button and enter the name in the DDE Service Window:



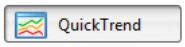
To edit a DDE service name, double-click it or select it and press the Edit. To remove a DDE service name, select it and press the Remove button.

It is currently possible to have more than one instance of Skkynet VINE, with one or more DDE service names in common, running on a single machine. If you plan to configure a system like this, you should ensure that each instance of VINE has unique data domain names. Otherwise, when any two of those VINEs are acting as servers, it is not possible to predict which one of them a given client will send data to.

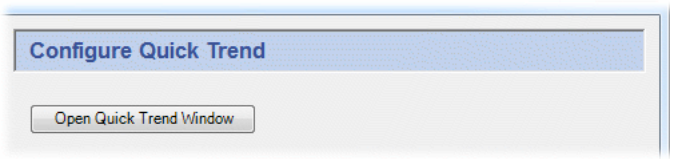
Non-English Characters in Excel

Checking this box will cause Excel to send strings of Unicode characters correctly, although slower than numerical data.

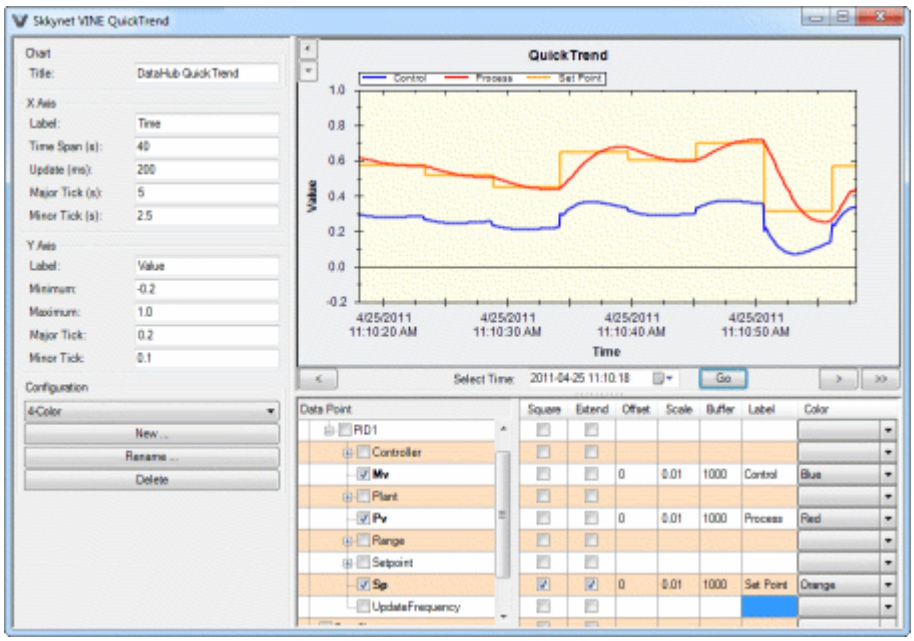
15.5. Quick Trend



The QuickTrend option lets you create a live trending graph for any number of data points in any domain of Skkynet VINE. You can configure the X and Y axes of the graph, zoom in on a particular area, and apply offsets and scales to raw data to plot widely disparate values together in a single chart.

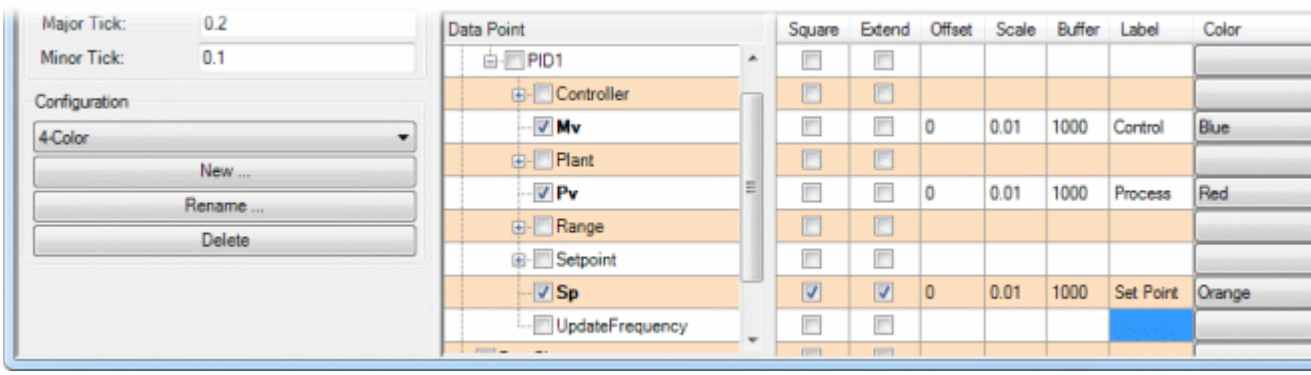


To configure a quick trend, press the Open quick trend window button:



Configuration

QuickTrend supports multiple display configurations, which you can manage using the Configuration options. The top button opens a dropdown list that contains all named configurations. The New..., Rename..., and Delete buttons let you create, rename, and delete configurations.



Data Points

You can select any number of data points to trend, from any data domain. The following options determine how the data will display in the chart.

Square

Removes interpolation of the line between two data changes, giving the plot a step-like appearance. This is useful for square functions.

Extend

When checked, this option extends the plot of a point's value as a straight line until the point changes value. With this option unchecked, no plot is shown until a point changes value. Then a straight line is plotted connecting the original value to the new value.

Offset

A value entered here will be added to each value of the point, creating an offset plot. This lets you view widely differing values in the same window.

Scale

A value entered here will be multiplied by each value of the point, creating an enhanced (or diminished if a fractional value) plot.

Buffer

This value determines how many data changes for this point will be stored in the local history, to allow for scrollbacks to review recently plotted data.

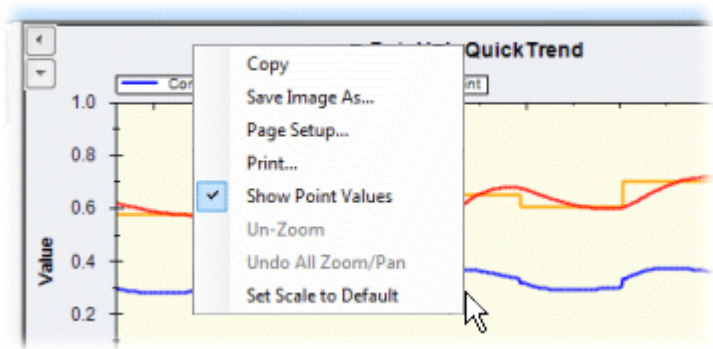
Label

Allows you to change the label for the point, whose default is the simple point name.

Working with the Display

There are a few features of the display that are not immediately obvious.

- You can scroll backwards and forwards through the history of the trend using the left and right arrow buttons, or choose a specific date and time with the calendar and time selector. The double-right arrow button returns the display to real-time trending.
- You can resize the trend display by dragging the gray borders on the left side or bottom. Move the mouse until you see a white, double-headed arrow, and drag.
- To zoom in to a part of the display, drag the mouse over the area that you want to zoom in on. To zoom back out and resume real-time trending, click on the Reset Pan/Zoom button at the bottom of the window.
- For other features, the QuickTrend display has a menu available on a right mouse-click.

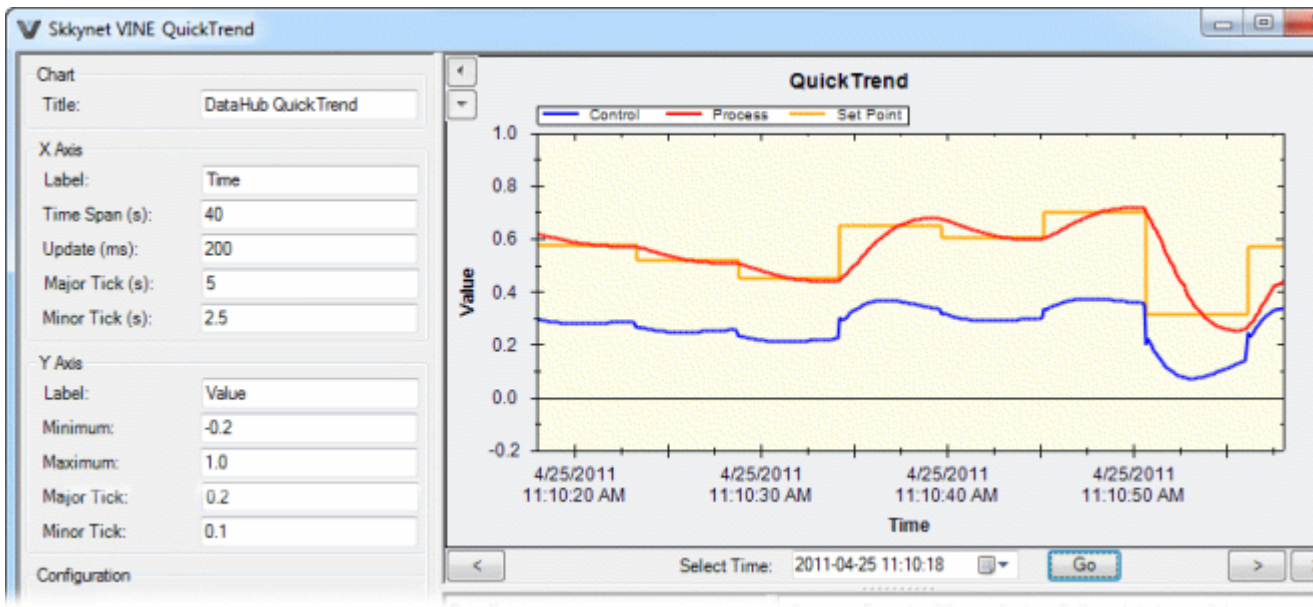


Using this menu you can copy, save, or print the current display, as well as unzoom and undo a zoom or pan. You can also hide or show point values, and reset the scale to the default.

Chart

Title:

The name of the chart, which will appear at the top.



X Axis

The X axis displays the time coordinate.

Label:

Any text string, displays at the bottom of the chart.

Time Span (s):

The total number of seconds that the chart will span.

Update (ms):

The update rate for the trend, in milliseconds.

Major Tick (s):

The time interval between major tick marks, in seconds.

Minor Tick (s):

The time interval between minor tick marks, in seconds.

Y Axis

The Y axis displays the value coordinate.

Label:

Any text string, displays at the far left of the chart.

Minimum:

The minimum value to include in the chart.

Maximum:

The maximum value to include in the chart.

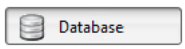
Major Tick:

The value between major tick marks.

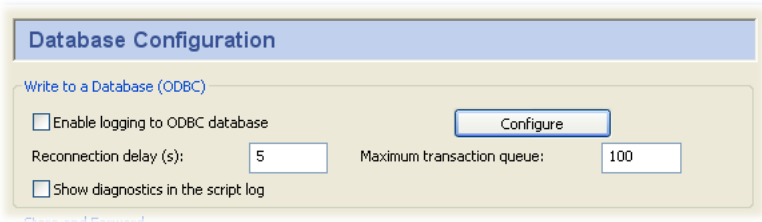
Minor Tick:

The value between minor tick marks.

15.6. Database



The Database option lets you configure VINE for writing data or making queries to any ODBC-compliant database.



Write to a Database (ODBC)

Enable logging to ODBC database

Globally enables or disables all ODBC logging activity.

Reconnection delay (s):

Specifies the number of seconds before a reconnect is attempted if the ODBC connection is broken.

Maximum transaction queue

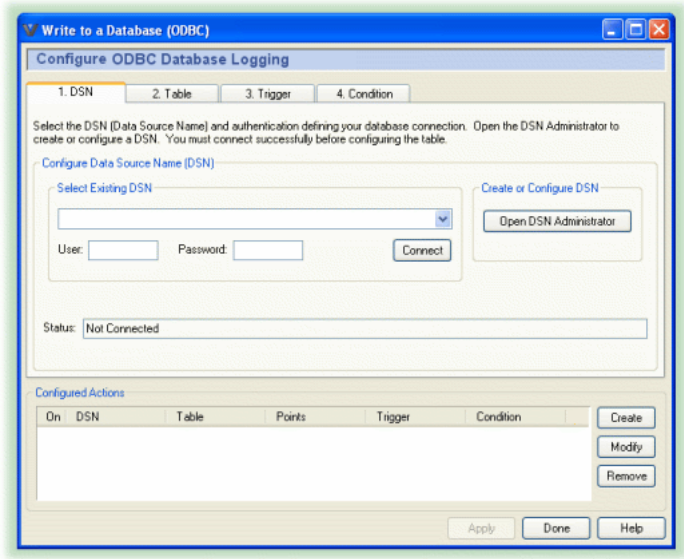
VINE maintains an in-memory queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. You can modify the depth of this queue to reduce the chance of involving the disk during busy periods. The queue depth for logging defaults to 100 messages.

Show diagnostics in the Script Log

Puts diagnostic messages about the connection in the Script Log.

Configure button

The Configure button opens the ODBC Data Logging Window.

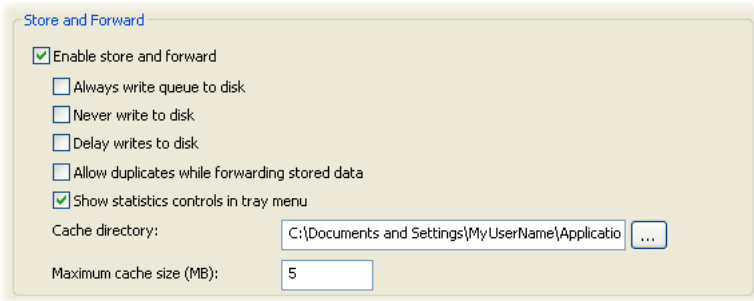


For detailed instructions on using this interface, please refer to Chapter 7, *Write to a Database*.

Store and Forward

The term *store and forward* refers to a type of database connection where the data is stored locally to disk and then later forwarded to the database. Skkynet VINE performs an advanced form of store and forward that only writes to disk if the database is not connected, or has been paused. If the database is available, the data will be transmitted directly to the database. This means that there is no penalty for using store and forward during normal operation. The VINE store and forward mechanism uses two levels of disk caching to ensure that all data gets logged, and nothing is lost.

When the database first becomes available after an outage, VINE starts writing cached values to the database, and continues writing new values to the cache. In this way, the values are inserted into the database in the order in which they are generated by the system. Once the cache is cleared, VINE then starts writing new values directly to the database.



Enable store and forward

Activates the store and forwarding feature.

Always write queue to disk

Data in the transaction queue will be written to disk cache first, and from there to the database. The safest protection against a crash is to check this box, and uncheck Delay writes to disk (below).

Never write queue to disk

The data in the transaction queue will be only stored in memory, and never written to disk.

Delay writes to disk

Data in the transaction queue will be written to disk at the most opportune times. The safest protection against a crash is to uncheck this box, and check Always write queue to disk (above).

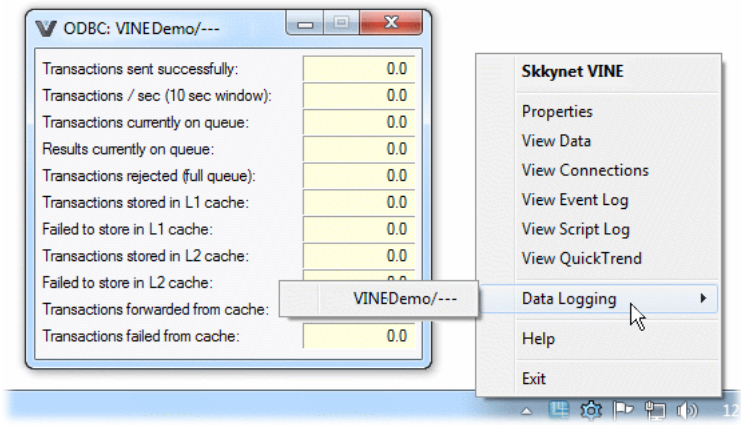
Allow duplicates while forwarding stored data

If the network breaks while transmitting data from a cache, Skkynet VINE needs to know how to handle any already-sent data when it reconnects. Leaving this box unchecked will require Skkynet VINE to track its cache position at all times, and modify that information each time a value is sent. This will impact the speed of every transmission, but it will ensure that no values get transmitted twice.

Checking this box will cause VINE to simply start from the beginning of the queue or cache on each reconnect, and retransmit some data. This significantly reduces data-handling complexity and decreases transmission rates. This option is particularly useful if network breaks are frequent and some duplication of logged data is acceptable.

Show statistics in tray menu

Adds a Data Logging entry to the VINE's system tray menu, which lets you open a statistics window:



Transactions sent successfully:

The number of transactions that were sent, either directly to the database, or to the disk cache.

Transactions / sec (10 sec window):

The sending rate for transactions, calculated over the past 10 seconds.

Transactions currently on queue:

The number of transactions in the queue.

Results currently on queue:

Not yet documented.

Transactions rejected (full queue)

The number of transactions that were rejected from the queue because it was full.

Transactions stored in L1 cache

The number of transactions taken off the queue and put into the first-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L1 cache

The number of transactions that were not able to be stored in the first-level cache.

Transactions stored in L2 cache

The number of transactions taken off the queue and put into the second-level cache. An internal algorithm determines which of the two caches is most appropriate for storing a given transaction.

Failed to store in L2 cache

The number of transactions that were not able to be stored in the second-level cache.

Transactions forwarded from cache

The total number of transactions forwarded from both caches. This number should be the sum of L1 and L2, once all transactions have been forwarded, and as long as VINE was started up with no cache on disk.

Transactions failed from cache

The number of transactions attempted from cache, could not be successfully delivered, and were stored for later transmission. This phenomenon may occur the first time that VINE learns that the database is not available. For example, you'll see this for every network break if you've checked Always write queue to disk.

Cache directory:

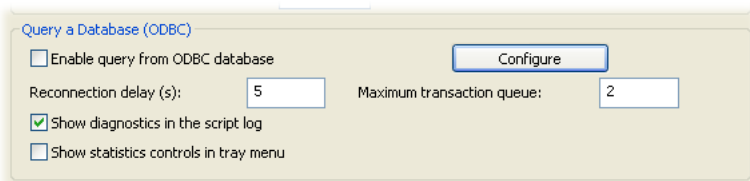
The path and directory name for the cache.

Maximum cache size (MB):

The amount of disk space to allocate for the cache, in megabytes.

Query a Database (ODBC)

In addition to writing data to a database, Skkynet VINE also allows you to query a database and put the resulting values into VINE.



Enable query from ODBC database

Globally enables or disables all ODBC query activity.

Reconnection delay (s):

Specifies the number of seconds before a reconnect is attempted if the ODBC connection is broken.

Maximum transaction queue

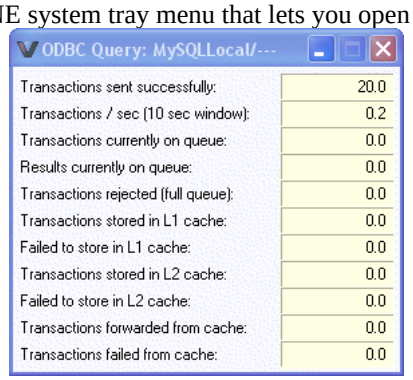
VINE maintains an in-memory queue of pending operations. This queue helps to avoid writing to disk during busy periods or during short database or network outages. You can modify the depth of this queue to reduce the chance of involving the disk during busy periods. The queue depth for queries defaults to 2 messages.

Show diagnostics in the Script Log

Puts diagnostic messages about the connection in the Script Log.

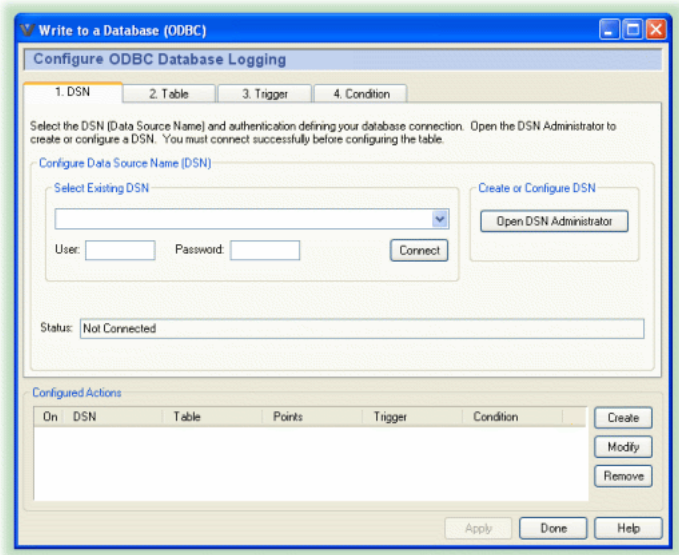
Show statistics controls in tray menu.

Creates a new entry in the VINE system tray menu that lets you open a statistics control window.


ODBC Query: MySQLLocal/---
Transactions sent successfully: 20.0
Transactions / sec (10 sec window): 0.2
Transactions currently on queue: 0.0
Results currently on queue: 0.0
Transactions rejected (full queue): 0.0
Transactions stored in L1 cache: 0.0
Failed to store in L1 cache: 0.0
Transactions stored in L2 cache: 0.0
Failed to store in L2 cache: 0.0
Transactions forwarded from cache: 0.0
Transactions failed from cache: 0.0

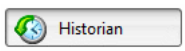
Configure button

The Configure button opens the ODBC Data Logging Window.

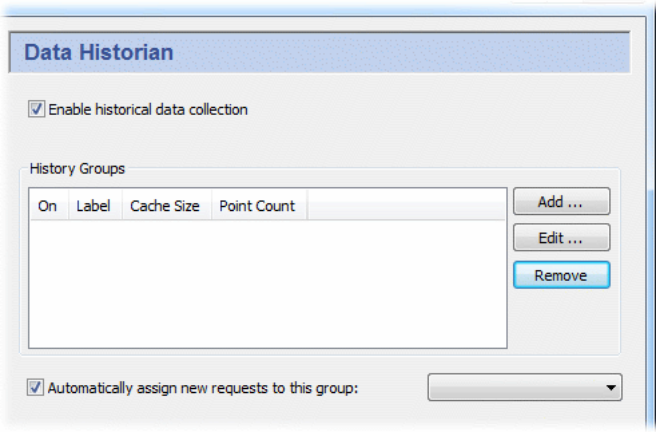


For detailed instructions on using this interface, please refer to Chapter 8, *Query a Database*.

15.7. Historian



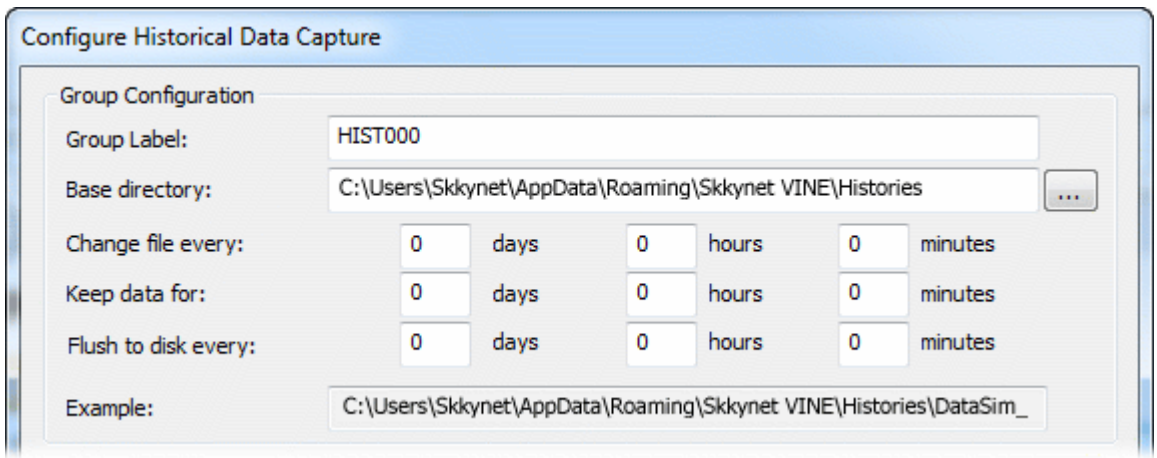
The Historian option allows you to collect and store histories for groups of data points. It gets configured automatically by the Quick Trend option, and can be configured manually as well.



Check the Enable historical data collection box to enable the Historian. Histories for one or more points are created and stored in groups. You can create as many groups as your system resources will allow, and activate the data collection for that group using its corresponding On check box in the list. To edit a history group, double-click it here, or select it and press the Edit button to open the Configure Historical Data Capture window (see below). To remove a group, highlight it and click the Remove button.

Group Configuration

To add a history group, press the Add button, which opens the Configure Historical Data Capture window:



Group label:

Any text string, used to identify the group. Skkynet VINE will assign a numbered label by default if nothing else is specified.

Base directory:

The directory in which the histories will be stored.

Change file every:

How often to close a recorded history file, and open another.

Keep data for:

How long to store data on disk.

Flush to disk every:

How often to flush the data from memory and write it to disk.

Example:

The directory tree and example filename that the history will be written to.

Deadband

A deadband is used to reduce the amount of data stored by only storing data if there is a significant change in value. This approach is superior to simply reducing the sampling frequency, which will lose information when data changes quickly, and will waste storage by saving the same values when data doesn't change. The deadband approach defines a resolution below which changes in data are deemed to be 'noise' and therefore ignored.

Absolute change:

Sets the deadband range to a single value. Any new value differing from the *baseline* (current value) by less than the number entered here will not be recorded. If a value's difference from the baseline is greater than or equal to this number, that value gets recorded and it becomes the new baseline for the absolute change deadband.

Percent change:

Sets the deadband range to be a percent of the last logged data value. Every new value is compared to that value, and if their percentage difference is less than the number entered here, the new value will not be recorded. If the difference is greater than or equal to this number, the value is recorded and that value becomes the new baseline for the percent change deadband.

If absolute change and percent change are used together there is an AND relationship between them. The Historian will ignore any value falling within either deadband. Only those values falling outside all deadbands (or equal to the outermost) will be recorded.

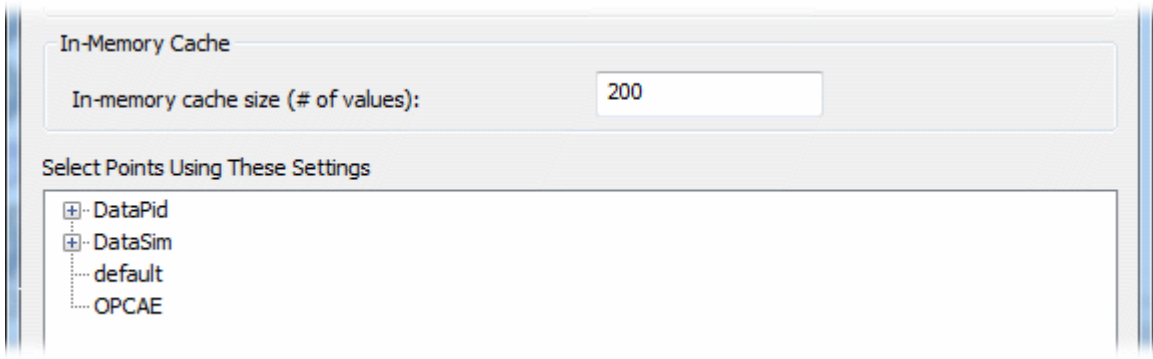
Maximum time between values:

Sets the maximum time period (in seconds, a real number) to deadband. When the maximum time is reached, the next new value received will be recorded, even if its value doesn't exceed the deadband. Note that the system does not automatically generate and insert a value when the maximum time is reached; it waits for the next new value. Whenever a new value is recorded the time calculation is restarted. If the maximum time parameter is not used, there is no time limit on how many values are ignored within a deadband.

Maximum number of skipped values:

Sets a maximum number of values to skip for the deadband. When the maximum number is reached, the next new value will be recorded, even if it doesn't exceed the deadband. Whenever a new value is recorded the countlimit is restarted. If the countlimit parameter is not used, there is no count limit on how many values are ignored within a deadband.

In-Memory Cache



In-memory cache size (# of values):

Determines the amount of memory allocated to storing values. This is specified by entering a number of values.

Select Points Using These Settings

Any number of points can be selected for the group, using the selection tree.

Click the OK button to save the group configuration and return to the Historian section of the Properties window. Then click the OK or Apply button in the Properties window to ensure t

Querying Data in the Historian

It is possible to query the Historian to extact raw data as well as statistics like minimums, maximums, averages, variances, standard deviations and so on over a given time period. This is done with a script that accesses the `Historian` class, explained in detail with an example in the VINE Scripting manual.

Historian File Format

The format for the Historian files consists of consecutive records as follows:

timestamp

An 8-byte floating point number indicating the number of seconds since January 1, 1970 00:00:00, UTC time, in IEEE floating point format.

value

An 8-byte floating point number, in IEEE floating point format.

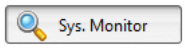
quality

A 2-byte integer number encoding OPC quality, in little-endian order (Intel byte order):

```
OPC_QUALITY_MASK = 0xc0;
OPC_STATUS_MASK = 0xfc;
OPC_LIMIT_MASK = 0x3;
OPC_QUALITY_BAD = 0;
OPC_QUALITY_UNCERTAIN = 0x40;
OPC_QUALITY_GOOD = 0xc0;
OPC_QUALITY_CONFIG_ERROR = 0x4;
OPC_QUALITY_NOT_CONNECTED = 0x8;
OPC_QUALITY_DEVICE_FAILURE = 0xc;
OPC_QUALITY_SENSOR_FAILURE = 0x10;
OPC_QUALITY_LAST_KNOWN = 0x14;
OPC_QUALITY_COMM_FAILURE = 0x18;
OPC_QUALITY_OUT_OF_SERVICE = 0x1c;
OPC_QUALITY_WAITING_FOR_INITIAL_DATA = 0x20;
OPC_QUALITY_LAST_USABLE = 0x44;
```

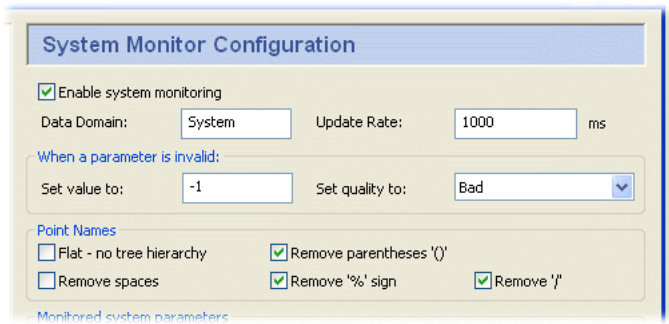
```
OPC_QUALITY_SENSOR_CAL = 0x50;
OPC_QUALITY_EGU_EXCEEDED = 0x54;
OPC_QUALITY_SUB_NORMAL = 0x58;
OPC_QUALITY_LOCAL_OVERRIDE = 0xd8;
```

15.8. System Monitor



The System Monitor option allows you to access any system performance data item, such as CPU usage, memory usage, process ID, disk space, network traffic, etc. with Skkynet VINE.

For example, by monitoring process ID you could determine whether a particular process is running or not. Any information accessed here becomes part of VINE's data set, and can thus be tunneled across the network, used in scripts or as email triggers, viewed in a spreadsheet, or stored in a database.



To enable system monitoring, check the Enable system monitoring box. There are several configuration options.

Data Domain:

The name of any VINE data domain. The values retrieved from the system will be shown as points in this data domain.

Update Rate:

The frequency that the system is polled and all selected points are updated. The minimum polling time is 100 ms., so the value entered here cannot be less than `100`.

A high update rate (a low number here) for many data points could use a great deal of CPU.

When a parameter is invalid

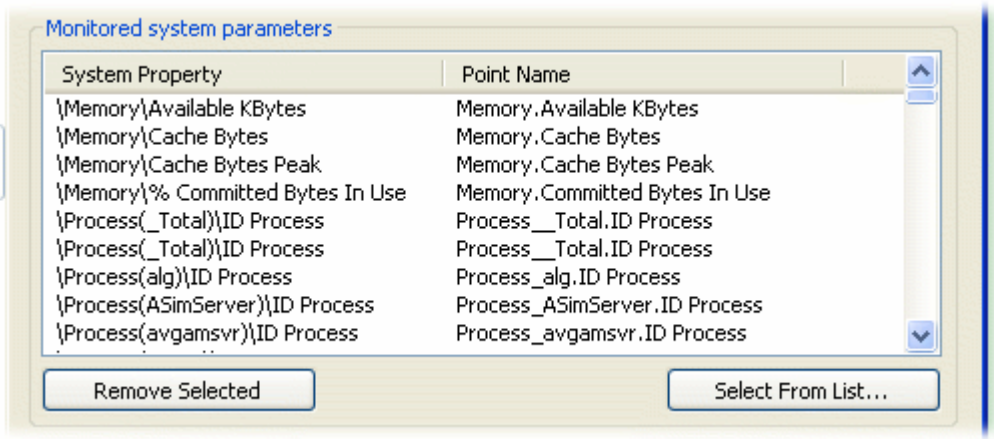
A parameter will be invalid if the object being monitored is not available. For example, if a process is not running then the parameters for that process will all be invalid. This is a useful way to monitor a system process or other object. For example, you could use a script or other client to watch a process ID, and when the process ID becomes `-1` you could generate an alarm indicating that the process is no longer running.

Point Names

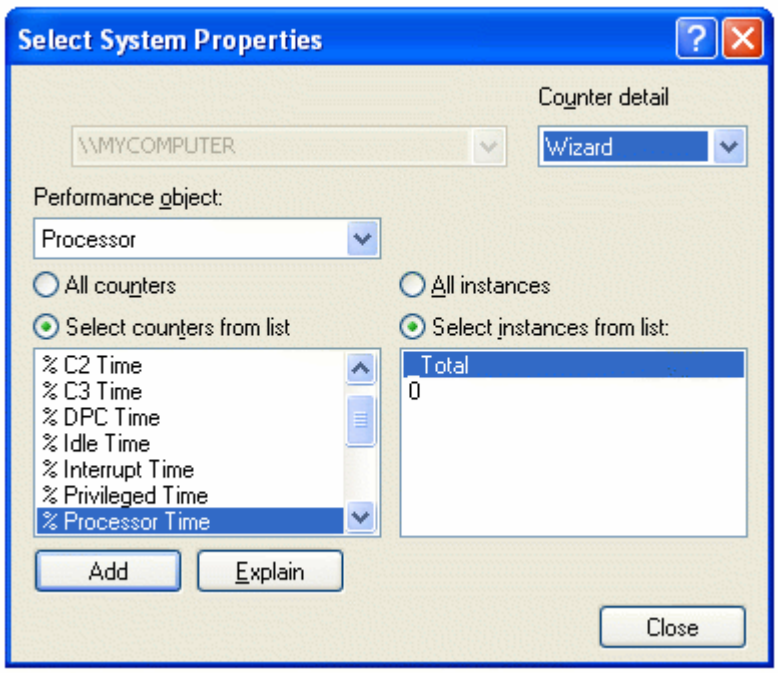
The System Monitor automatically creates Skkynet VINE point names based on the names of the system properties. Some client programs cannot work with point names containing special characters. This

section allows you to specify which characters will be removed from the property name when constructing the point name.

Monitored system parameters



This list shows the system properties you have chosen, and their corresponding point names in VINE. To add names to the list, click the Select From List button. This will open the Select System Properties dialog:



Depending on your system, this dialog may take a few seconds to appear. If it does not come up, the Event Log will contain a message. Otherwise, just be patient, it will open eventually.

In the Select System Properties dialog you can specify which items to add to your list of monitored system properties, according to these criteria:

- **Performance object:** A list of all available objects, such as CPU, Memory, Process, Print Queue, TCP, etc.

- **Counters:** All of the available data categories related to the selected performance object. You can choose all counters, or select specific counters from the list. The Explain button opens a window with an explanation of the selected counter.
- **Instances:** All of the instances of the chosen performance object. For example, if you chose Process for your performance, this list will show all of the processes running on your system. You can choose all processes or select specific processes from the list.

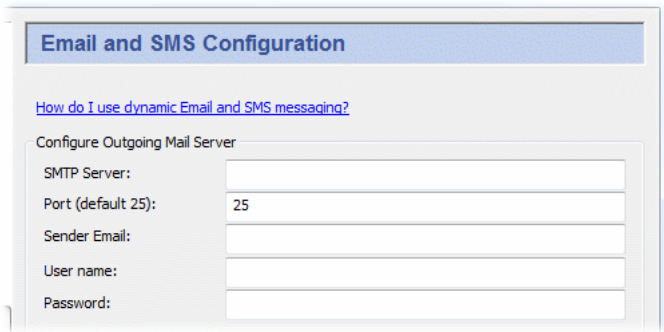
  A number in this list normally indicates a selection from multiple objects of a given type, and `_Total` means the total across all of the objects. For example, if you are looking at `Processor` in a multi-processor machine, you will see a number (`0`, `1`, etc.) for each processor and a `_Total` for the cumulative statistic over all processors.

1. Select a performance object, and counters and instances as applicable.
2. Click the Add button to add the selected items to the Monitor system parameter list in the VINE Properties window.
3. Click the Apply or OK button in the Properties window when you are finished making your choices and filling the list, to apply your changes. You should be able to view the results in the Data Browser.

If you change your mind on what points to monitor, you can change the list at any time. Any points you remove from the list will continue to exist in VINE until it is shut down and restarted. Please refer to Section 13.1, *Data Points* for more information on creating and deleting points.

15.9. Email/SMS

The Email/SMS option lets you configure Skkynet VINE to send emails and SMS text messages. The outgoing mail server is configured once, and each email message is configured separately.



Configure Outgoing Mail Server

SMTP Server:

The name of the SMTP server. You can use the same SMTP server listed in your email client program.

Port (default 25):

The SMTP port number (typically this is port 25).

Sender Email:

The email address of the sender. This will appear in the `From` field of the email.

User name:

The log-in name you use to access this SMTP account.

Password:

The applicable password.

Security

Security
Never attempt to connect securely via SSL
Always use SSL (fail if unavailable)
Automatically select most secure connection
Accept invalid or untrusted certificates

Never attempt to connect securely via SSL

Ensures that SSL is not used.

Always use SSL (fail if unavailable)

Ensures that SSL is always used, and that any connection attempt without it will fail.

Automatically select most secure connection

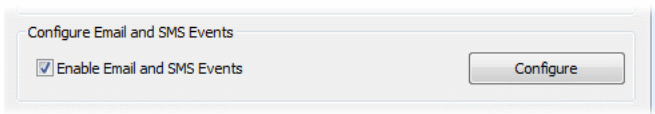
Allows the VINE to choose the most secure type of connection available on the mail server.

Accept invalid or untrusted certificates

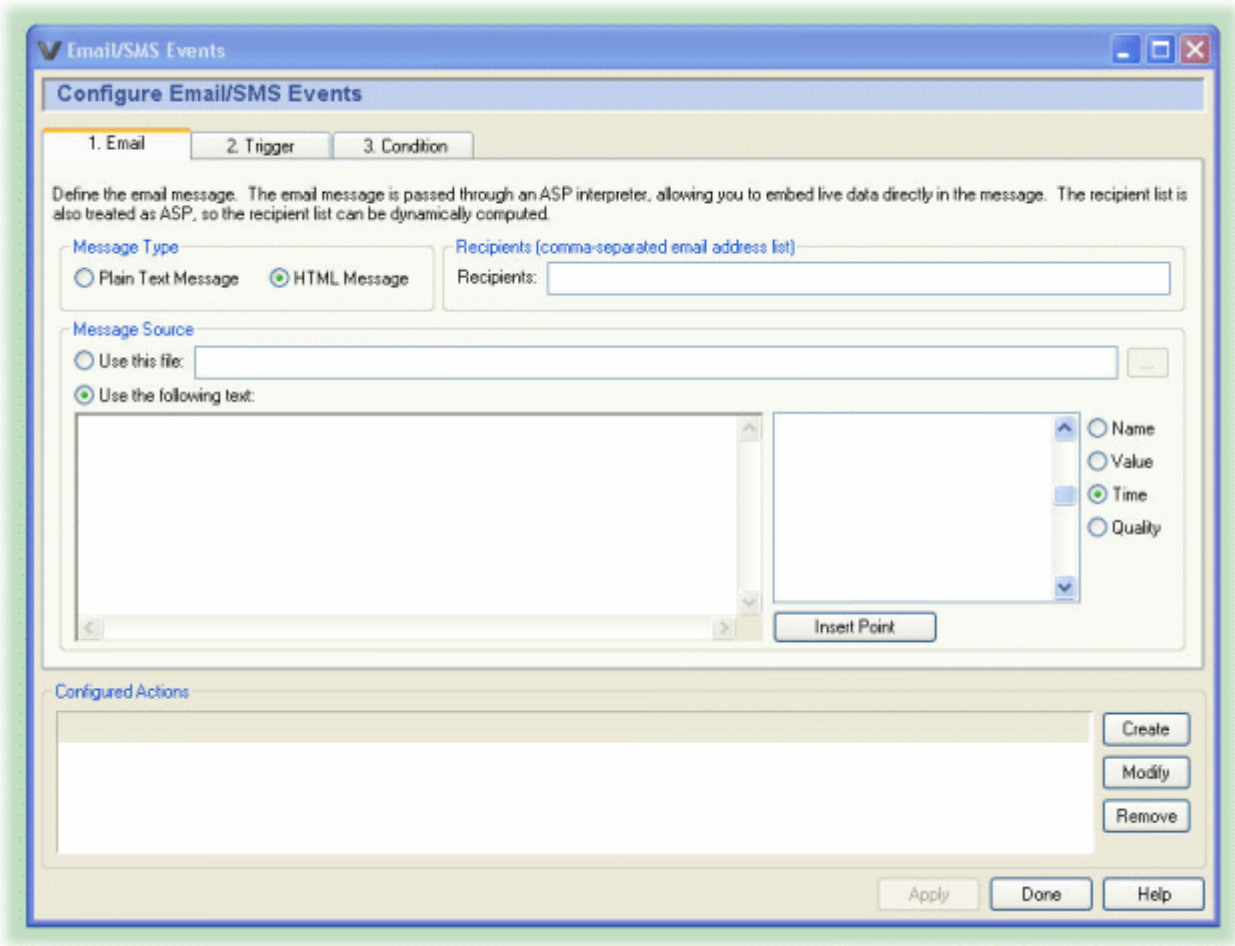
Ignores security certificate warnings.

Click the OK button to submit your entries.

Configure Email and SMS Events



To Configure Emails press the Configure button to open the Email/SMS Events window:



A complete explanation for this window can be found in Chapter 10, *Email and SMS*, starting with the Sending a Test Message.

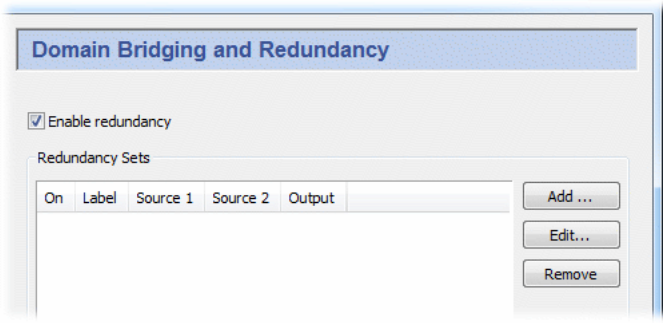
Click the OK button to submit your entries.

15.10. Redundancy

The Redundancy option lets you configure redundant connections to Skkynet VINE.

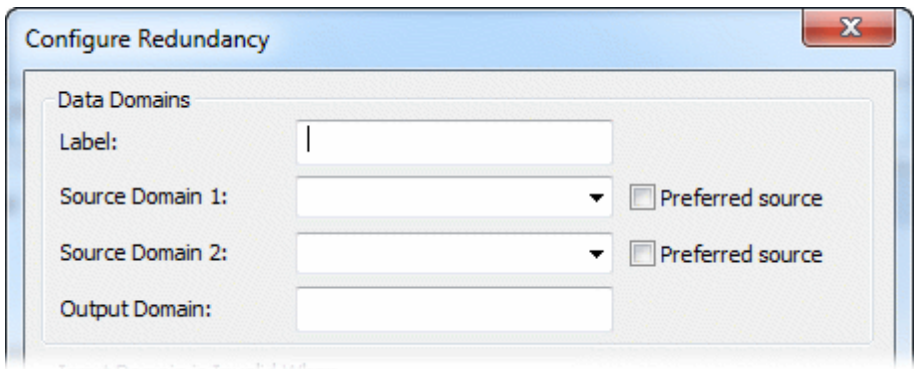
The purpose of redundancy is to collect data from two data sources and present to the client program a single output data set. Skkynet VINE will determine which source will be presented to the client program, and switch between the two sources without affecting the client. The client will only read data from the output data set.

The two input and one output data sets are maintained as separate data domains in Skkynet VINE. The sources do not need to be the same protocol, so redundancy can be applied to two sources, for example where one is a direct OPC connection and the other is a tunnel.



Check the Enable redundancy box to activate this feature. Redundant connections are created and stored in sets. You can create multiple redundant sets, and activate or deactivate each set using its corresponding On check box in the list. To edit a redundancy set, double-click it here, or select it and press the Edit button to open the Configure Redundancy window (see below). To remove a set, highlight it and click the Remove button.

To create a redundancy set, press the Add button, which opens the Configure Redundancy window:



Data Domains

Label:

A name used by Skkynet VINE to identify the redundancy set. There should be no spaces in the label. It doesn't matter what label is chosen, but it should be unique to other labels.

Source Domain 1:

The VINE data domain for the first data source. If this is the preferred source, check the Preferred source button.

If a preferred data source has been specified then VINE will use that source whenever possible, even if the other source is also available. This is useful if the two data sources have different characteristics. For example, the preferred source may offer a higher bandwidth than the other source. If neither data source is selected as preferred, VINE will maintain whichever data source is currently being used until it meets any invalid criteria (see below).

Source Domain 2:

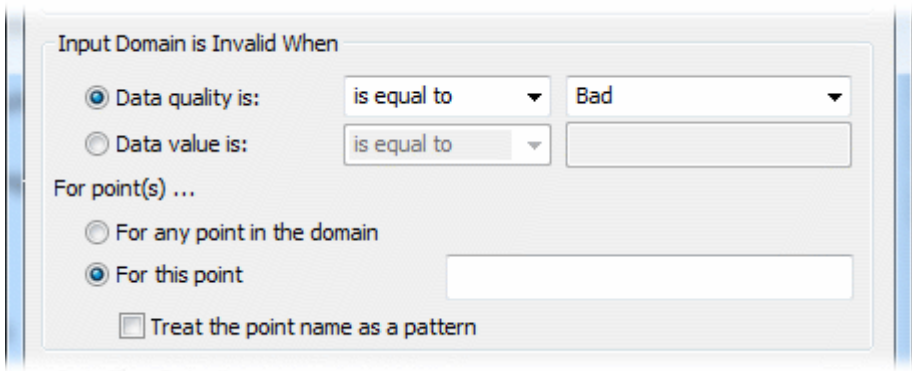
The VINE data domain for the second data source. If this is the preferred source, check the Preferred source button.

Output Domain:

A name for a VINE data domain which will be the output of the redundant connection, to which the client will connect. If the output domain does not exist, VINE will create it.

Input Domain is Invalid When

Entries in this section determine when VINE should switch from one redundant data source to the other.



Data quality is:

Gives you the option of switching data sources based on a change in data quality for the point(s) you have selected (below). You can set the critera of equal to or not equal to a list of available qualities, such as:

Bad	EGU Exceeded	Last Usable	Sensor Calibration
Comm Failure	Good	Local Override	Sensor Failure
Config Error	Initializing	Not Connected	Sub Normal
Device Failure	Last Known	Out of Service	Uncertain

Generally speaking, all of the above qualities are considered not good except for Good and Local Override. To ensure that you are getting good quality data from your OPC server, you can switch when Data quality is not equal to Good.

Data value is:

Gives you the option of switching data sources based on a change in the value of the data point(s) you have selected (below).

For point(s)

Allows you to select which points you want to monitor for quality or value (see above).

For any point in the domain

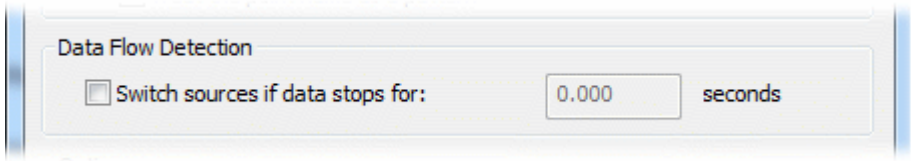
Lets you monitor all points in the domain and switch when any one of them meets the criteria.

For this point

Lets you specify an individual point name. The point name can be applied to a single point, or to a group of points whose names match the pattern.

Data Flow Detection

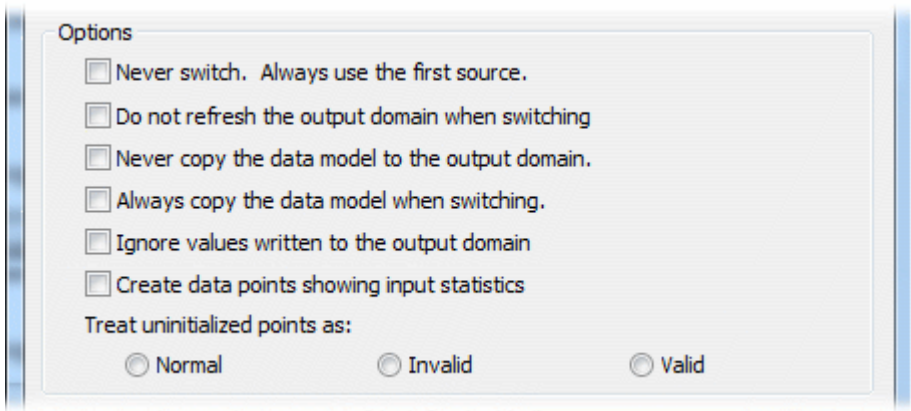
For data that changes regularly, Data Flow Detection lets you switch data sources whenever a gap in the data flow is detected. This option will watch for any change in any data point from an input domain. If any point in the input domain changes within the specified time, the entire input domain is assumed to be available. A data change must also pass the validity checks above in order to be considered valid.



Switch sources if data stops for:

The number of seconds that the data flow from the domain must stop before VINE will switch to the redundant data domain.

Options


Options
Never switch. Always use the first source.
Do not refresh the output domain when switching
Never copy the data model to the output domain.
Always copy the data model when switching.
Ignore values written to the output domain
Create data points showing input statistics
Treat uninitialized points as:
Normal
Invalid
Valid

Never switch. Always use the first source.

If you never switch, you never use a second source domain. Thus, this allows you to effectively create a copy of a domain, by copying Source Domain 1 into the Output Domain. To have the output domain function as a read-only copy of the source domain, select the Ignore values written to the output domain. option, as explained below.

Do not refresh the output domain when switching.

Keeps the existing data in the output domain during a switch, and only updates with values from the new domain when a change occurs. Choosing this option may cause a mismatch between input and output domains for an indeterminate length of time.

Normally during a switchover VINE copies all values from the new source domain into the output domain. This copying activity might cause delays in updating the output domain for extremely large numbers of points. If that is your situation, and you know that your two servers are synchronized in all meaningful ways, you may wish to select this option. However, you need to keep in mind that values in the input and output domains may not match for an undetermined period of time.

Never copy the data model to the output domain.

Preserves the data model of the output domain, or if there is no data model, flattens the data model from the input domain. In either case, the data point names are maintained. This can be helpful if targeting a system with limited system resources, such as an embedded system, or if you have an existing data model on the output domain and do not want it overridden by the data model on the input domains.

Always copy the data model when switching.

Normally the output domain tracks changes in the data model, and when a switch occurs, if the data model has changed, it gets rewritten. This option forces the output domain to copy the data model, whether it has changed or not.

Ignore values written to the output domain.

Normally, data written to the output domain propagates back to the input domains. This option prevents that from happening. Data written to the output domain will not be written to the input domain. Used with Never switch. Always use the first source. (above), this will make the output domain function as a read-only copy of the input domain.

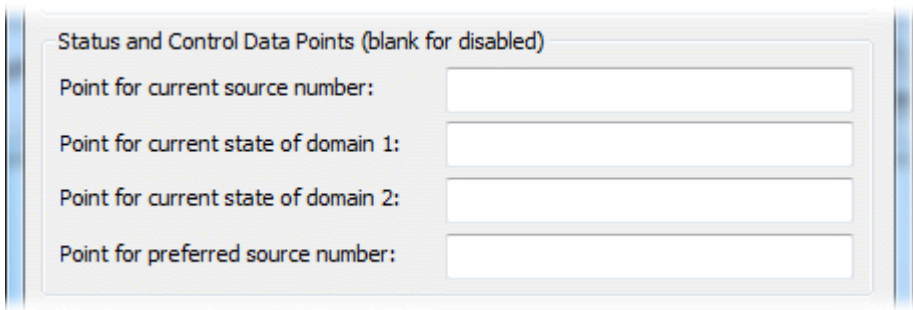
Create data points showing input statistics

Used for debugging, this option creates extra points in the root of the output domain that indicate how many points are considered valid, invalid and uninitialized for each input domain.

The radio buttons under Treat uninitialized points as: let you choose whether a point in an input domain that has never been assigned a value (BAD quality, `0` timestamp, and `0` value) should be treated as:

- Normal: The validity rules apply normally to it.
- Invalid: The domain will never be used as an input until all data points have a value assigned to them.
- Valid: Any uninitialized points are ignored when determining whether the input domain is valid.

Status and Control Data Points (blank for disabled)


Status and Control Data Points (blank for disabled)
Point for current source number:
Point for current state of domain 1:
Point for current state of domain 2:
Point for preferred source number:

Point for current source number:

The name of a VINE point that will indicate which source is in use.

Point for current state of domain 1:

The name of a VINE point that will indicate the state of Domain 1.

Point for current state of domain 2:

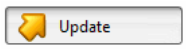
The name of a VINE point that will indicate the state of Domain 2.

Point for preferred source number:

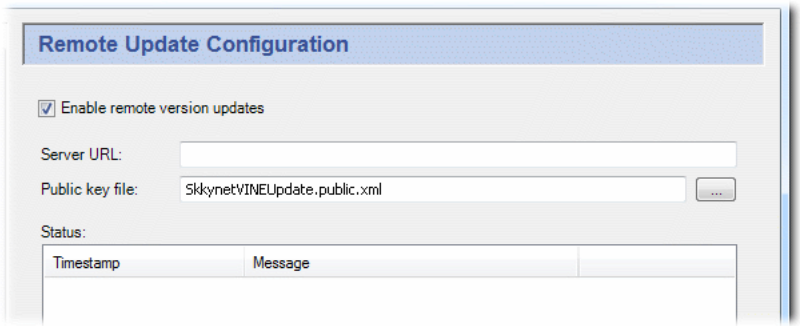
The name of a VINE point that will indicate which data source is your preferred source.

Click the OK button to submit your entries.

15.11. Update



The Update option allows you to update Skkynet VINE from a remote location or server without downloading the software. This is particularly convenient for large-scale distributed systems.

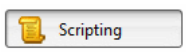


To do a remote update, follow these steps:

1. Ensure that Skkynet VINE is running with Administrator permissions.
2. Check the Enable remote version updates box to enable the Update feature.
3. In the Server URL field, enter the URL for the location of the new version, as provided by your system administrator or other authorized person.
4. In the Public key file field, enter the public key, as provided by your system administrator or other authorized person.
5. Press the Apply button.
6. Press the Update button to initiate the update.

   You will see some messages scroll past in the Status window, and then Skkynet VINE will restart, running the updated version.

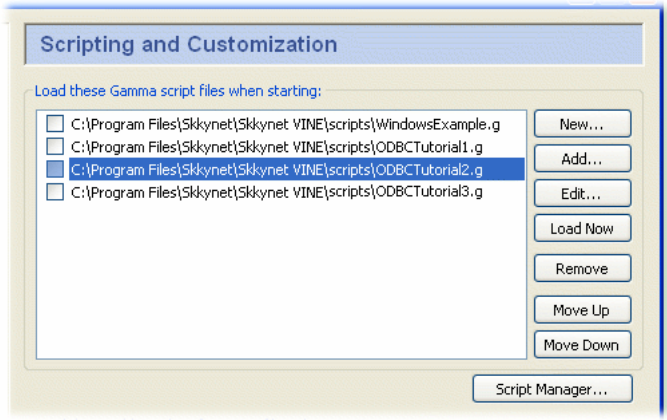
15.12. Scripting



The Scripting option lets you write, edit, and run scripts, as well as work with configuration files. Please refer to the DataHub Scripting manual for more details.

Load these Gamma script files when starting:

Here you can create and access Gamma scripts.

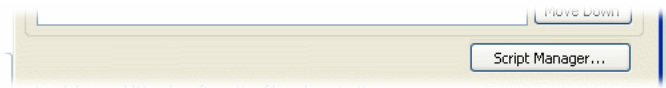


The DataHub Scripting manual has a more complete explanation for these options, but briefly they are as follows:

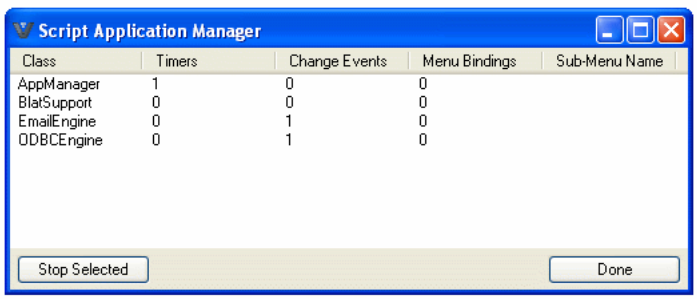
- New creates a new script from parameters you specify.
- Add lets you add a script to the list.
- Edit opens the script in the Script Log for editing.
- Load Now loads the selected script into the Gamma interpreter for immediate processing.
- Remove deletes the selected script from the list.
- Move Up and Move Down move the selected script up and down in the list

Script Manager

VINE's Script Application Manager let you view a list of scripts and stop a running script. To access it, press the Script Manager button.



This will open the Script Application Manager window:



To stop a script, highlight it, and press the Stop Selected button.

The columns display the following information:

- Class: the name of the instance of the Application class created in the script.
- Timers: the number of timers active in this script.

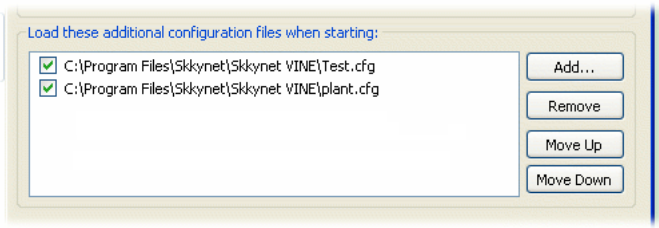
- Change Events: the number of change events active in this script.
- Menu Bindings: the number of menu entries that this script has placed in the system tray menu.
- Sub-Menu Name: the name of the submenu in the system tray menu into which this script has placed its menu entries.

Load these additional configuration files when starting:

Configuration files let you start Skkynet VINE from a known configuration. Normally there is one configuration file that comes with the VINE distribution. Each change you make in the Properties window gets written to that file automatically, and saved. However, it is also possible to create and/or edit configuration files with a text editor (see Section 1.5, *Configuration Files*).

Any additional configuration files must be listed here in order to be read by Skkynet VINE. The files in the list are used in order from the top down, with the last file taking precedence. That is, if a value is assigned to a variable from within a configuration file, its value will be changed by subsequent assignments, whether they come from within that config file or from any other.

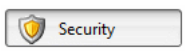
Creating or editing configuration files should only be attempted by experienced users.



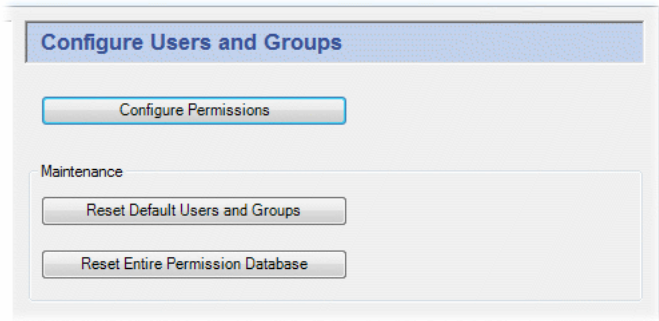
The Add button opens a standard file browsing window, from which you can choose configuration files to add. The Remove button removes the configuration file that is highlighted.

The Move Up and Move Down buttons rearrange the order of the configuration files. Order is important. These files are read from the top down; variables change with each subsequent assignment, taking on the last value assigned.

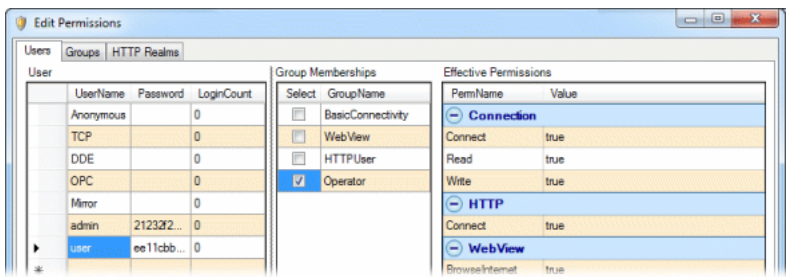
15.13. Security



The Security option lets you configure security for Skkynet VINE tunnel/mirror, TCP, OPC, and DDE connections. For more information on VINE security, please refer to Chapter 12, *Security*.



Click the Configure Permissions button to open the Edit Permissions window.

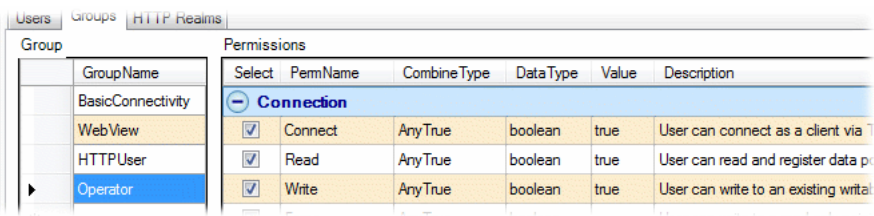


Here you can create and modify groups, and then assign users to those groups.

Groups

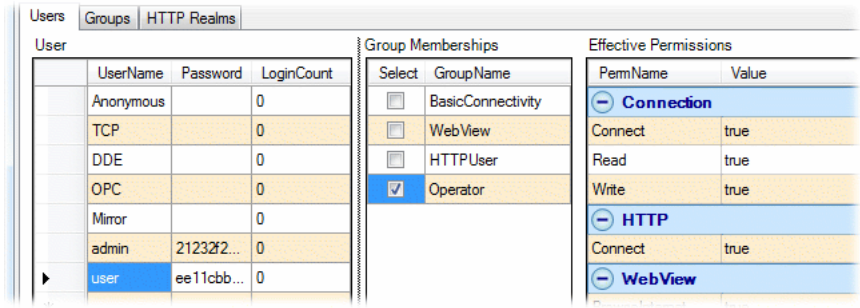
Groups provide a convenient way to configure a number of users who have identical permissions. Each group can be assigned a unique set of permssions from the Permissions table. There are three default groups: `Permissive`, `WebView`, and `HTTPUser`. To add a group, type a group name in the bottom row of the Groups table. Check or uncheck the boxes to assign permissions.

For example, in the illustration below an `Operator` has been added that has been given Connection permissions for `Connect`, `Read`, and `Write`.

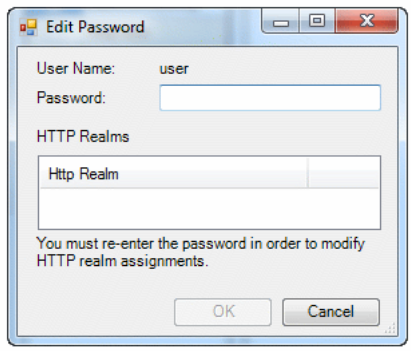


Users

There are two kinds of users--normal and special. Normal users correspond to individuals with a name and a password. Special users provide a way to offer different security models for different protocols. For more information on types of users, please refer to Section 12.3, *User Authentication*.



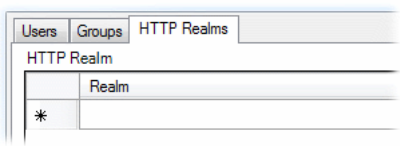
To add a user, type a user name in the bottom row of the User table. When you press **Enter**, a password dialog will appear:



Enter a password and select an HTTP realm for that user. When you click OK, a string of charaters will appear in the Password field for that user. Passwords are stored using a reasonably strong non-reversible encryption. If a user forgets his password, it is not recoverable. To change HTTP realm for a user, their password must be reentered. For more information on passwords, please refer to Section 12.6, *Passwords*.

HTTP Realms

Here you can maintain a list of HTTP authentication realms. This list is accessed by the VINE Web Server, as described here: .



To add an authentication realm, simply type it into the list. One or more of these realms are assigned to each user when their password is configured (see above).

Common Scenario

The most common Skkynet VINE security configuration is to allow any user to connect via OPC or DDE, while only allowing authorized users to connect via TCP or via a tunnel/mirror. This eliminates exposure of the the TCP and tunnel/mirror connections to unwanted Internet and network clients. OPC and DDE are not exposed in this way.

To configure this scenario, you need to remove all group memberships from the special `Anonymous`, `TCP`, and `Mirror` users. Simply click on each of these user names in turn, and uncheck all group memberships for that user. When you are finished, only `DDE` and `OPC` should have any group memberships.

Permissions

Connect

: This user is allowed to maintain a connection to Skkynet VINE. When a connection is made, the client has a 5-second grace period in which to authenticate before the client is disconnected. If the client does not have Connect permissions after the grace period expires, it will be disconnected.

Read and register points

: This user is allowed to read point values and subscribe to point value changes.

Change point values

: This user is allowed to write a new point value to Skkynet VINE.

Force value changes

: If the user has Change point values permission, he may also have this permission. In this case, the user will able to send the **force** and **cforce** commands to VINE, which will override the read-only status and timestamp check for a point, thereby forcing a write to succeed where it would otherwise fail.

Create new points

: This user is allowed to create new points in existing data domains in Skkynet VINE.

Delete an existing point

: This user is allowed to delete a point from Skkynet VINE.

Normally, no client should be allowed to delete points from Skkynet VINE. Deleting points can be very disruptive for existing clients. Use this permission with caution.

Create a new data domain

: This user is allowed to create new data domains. Normally you should also set Create new points permission when you set this permission for a user.

Load a configuration file

: This user is allowed to tell Skkynet VINE to load a specific configuration file.

Create and edit users and groups

: This user is allowed to create and edit users and groups non-interactively.

Change the program configuration

: This user is allowed to transmit commands to Skkynet VINE to alter VINE's configuration. This normally includes actions like enabling and disabling particular interfaces and functions within the cdh.

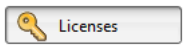
Change auto domain creation

: This user may change the flag indicating whether Skkynet VINE should automatically create a data domain when a user requests a point in a non-existent data domain.

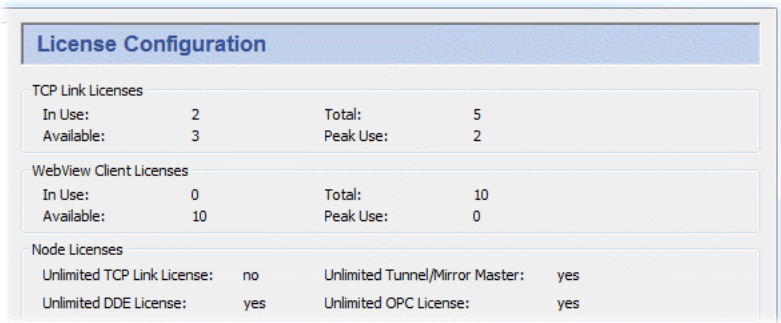
Shut down the program

: This user may transmit an **exit** command to Skkynet VINE, causing it to shut down.

15.14. Licenses



The Licenses option lets you view and install licenses for Skkynet VINE. When Skkynet VINE starts up it will run in demo mode (one hour time limit) if no licenses are found. If any license is found, Skkynet VINE switches to license mode, and each connection then requires a license.



TCP Link Licenses

Each simultaneous TCP connection (except for the Tunnel/Mirror Master) to VINE requires one TCP Link License.

In Use

How many TCP Link Licenses are currently being used.

Available

How many are not being used and are still available.

Total

The total number of TCP Link Licenses installed.

Peak Use

The highest number of TCP Link Licenses that have been used since the beginning of the current session.

WebView Client licenses

Each simultaneous WebView client connection requires one WebView Client License.

In Use

How many WebView Client licenses are currently being used.

Available

How many are not being used and are still available.

Total

The total number of WebView Client licenses installed.

Peak Use

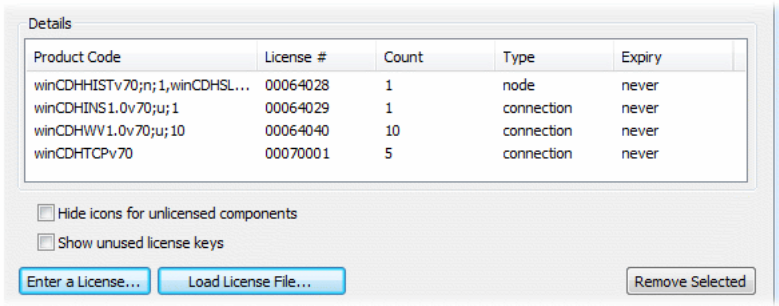
The highest number of WebView Client licenses that have been used since the beginning of the current session.

Node Licenses

This box displays which unlimited licenses are currently available for Skkynet VINE: TCP, DDE, Tunnel/Mirror, and OPC.

Details

The Details list shows the licenses currently installed.



There are two general viewing options for licenses:

Hide icons for unlicensed components

: Completely removes the greyed-out property icons for VINE features that are not licensed.

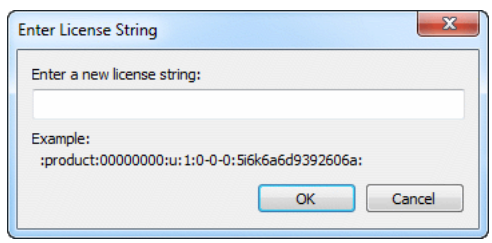
Show unused license keys

: It is possible that a license file may contain upgrade licenses for several different versions or instances of a VINE. If one or more license files have been loaded, checking this box will show in the Details list all the licenses available, including those that might be for older versions or other installations of VINE.

Installing Licenses

Licenses can be entered individually or loaded from a file.

- The Enter a License... button opens the Enter License String window:



  Here you can paste or manually enter the text string for the license provided by Cogent. Make sure to include all colon (`:`) characters in the string.

  The license string may contain the characters `l` and `1` which can look nearly identical in some type fonts. If possible, it is best to copy and paste the string, rather than retyping it.

- The Load License File... button opens a Windows file selection window. Browse to find the directory and license file that you want to load. License files end with a `.lic` extension. Once you have found

the license file, click the Open button to load the file. (Please refer to Configuration and License File Locations in Section 1.5, *Configuration Files* for more information on license file locations.)

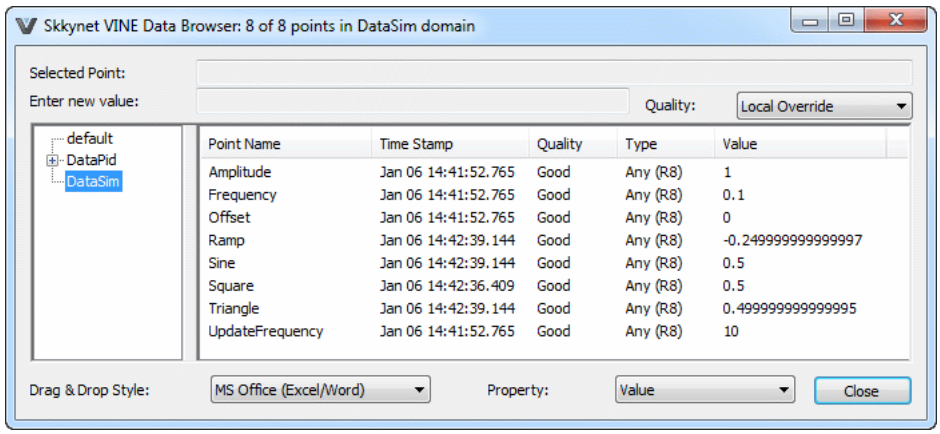
To remove a license from your system, select one or more licenses in the Details window, then click the Remove Selected button.

Chapter 16. Other Windows and Programs

16.1. Data Browser

This window gives a real-time view into Skkynet VINE. You can open this window in either of two ways:

- Click the View Data button in the Properties window.
- Right click the VINE icon in the system tray and select View Data from the pop-up menu.

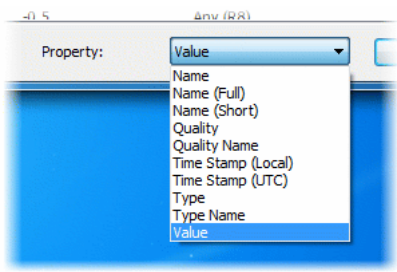

Skkynet VINE Data Browser: 8 of 8 points in DataSim domain
Selected Point:
Enter new value:
Quality:
Local Override
default
DataPid
DataSim
Point Name
Time Stamp
Quality
Type
Value
Amplitude Jan 06 14:41:52.765 Good Any (R8) 1
Frequency Jan 06 14:41:52.765 Good Any (R8) 0.1
Offset Jan 06 14:41:52.765 Good Any (R8) 0
Ramp Jan 06 14:42:39.144 Good Any (R8) -0.249999999999997
Sine Jan 06 14:42:39.144 Good Any (R8) 0.5
Square Jan 06 14:42:36.409 Good Any (R8) 0.5
Triangle Jan 06 14:42:39.144 Good Any (R8) 0.499999999999995
UpdateFrequency Jan 06 14:41:52.765 Good Any (R8) 10
Drag & Drop Style:
MS Office (Excel/Word)
Property:
Value
Close

All data domains in VINE are shown in the tree on the left, and all the points of the selected data domain are listed on the right. Clicking on a point name selects it, and puts its name into the Selected Point field at the top of the window. A snapshot of the value of the point at the time you clicked appears in the Enter new value field. You can change the value of the point by entering a new value in this field (i.e. type in the value and press the **Enter** key). You can also drag the column headers to change the order of the columns.

Drag and Drop Style and Property

There are several options for the drag and drop style, depending on the program in which you want to place the data. In addition, there is a Snapshat (Plain Text) option that allows you to put in labels and data properties that don't change.

The Data Browser also provides a way to select the Property of a data point to drag and drop, such as timestamp, quality, and point type.



The following properties are available:

Name

The name of the point, including its data hierarchy of assemblies, subassemblies, attributes, and properties, if any. For example:

```
PID1.Range.Amplitude
```

Name (Full)

The name of the point, including its data hierarchy and also domain name. For example:

```
DataPid:PID1.Range.Amplitude
```

Name (Short)

The short name of the point, without any of the data hierarchy. For example:

```
Amplitude
```

Quality

The quality of the point, an integer.

Quality Name

A text string that corresponds to the integer value of the quality of the point.

Time Stamp (Local)

The local time stamp, in seconds. In Excel, you can format this using a custom format for the cell. For example, to display the data and time to the nearest millisecond in a worksheet cell, you can use `m/d/yyyy h:mm:ss.000` as your custom format. Note that the milliseconds use a dot, not a colon.

Time Stamp (UTC)

The UTC time stamp, in seconds. You can format this in Excel using a custom format, as explained above.

Type

The type of the data contained in the point, as an iteger.

Type Name

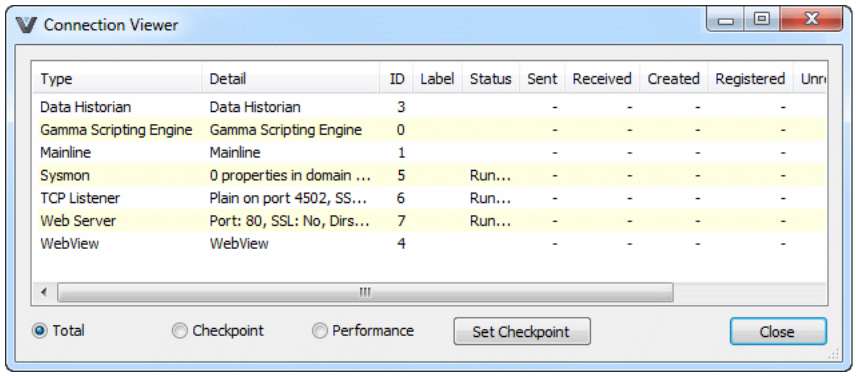
A text string that corresponds to the integer value of the data type.

Value

The value of the point.

16.2. Connection Viewer

This window gives a real-time view into all Skkynet VINE connections. You can open this window by clicking on the View Connections button in the Properties window.



The various columns identify the connection and show its status, with connection statistics, as follows:

- **Sent -** point changes sent from VINE to the connecting program.
- **Received -** point changes sent from the connecting program to VINE.
- **Created -** data points created in VINE by this connection.
- **Registered -** data points registered in VINE by this connection.
- **Unregistered -** data points unregistered in VINE by this connection.
- **Dropped -** data point changes that Skkynet VINE attempted to send, but could not, because the client was too busy to receive them before the next point change. When VINE drops a point change, it only drops values that have already been superseded by a newer value. Skkynet VINE will never drop the latest value.
- **Blocked and Unblocked -** If Skkynet VINE identifies that the client is unable to keep up with the transmission data rate, it will mark the client as *blocked*, indicating that the client will receive no new value changes until it is able to cope with them. VINE *unblocks* the client after a short period of time, or when it identifies that the client has queue space to accept new data changes.

  If a client is blocked, superseded point changes will be dropped. The latest value for each point is queued so that when the client is unblocked it will receive the latest values for all points that changed while it was blocked. Again, Skkynet VINE never drops the latest value for any point in which a client is interested.

- **CPU** - the CPU time in seconds that a given thread has consumed since it started. When the Performance option is selected, the CPU percentage is the average percentage of the CPU resources that the thread has been using since the Connection Viewer window was opened, or the Set Checkpoint button was pressed.

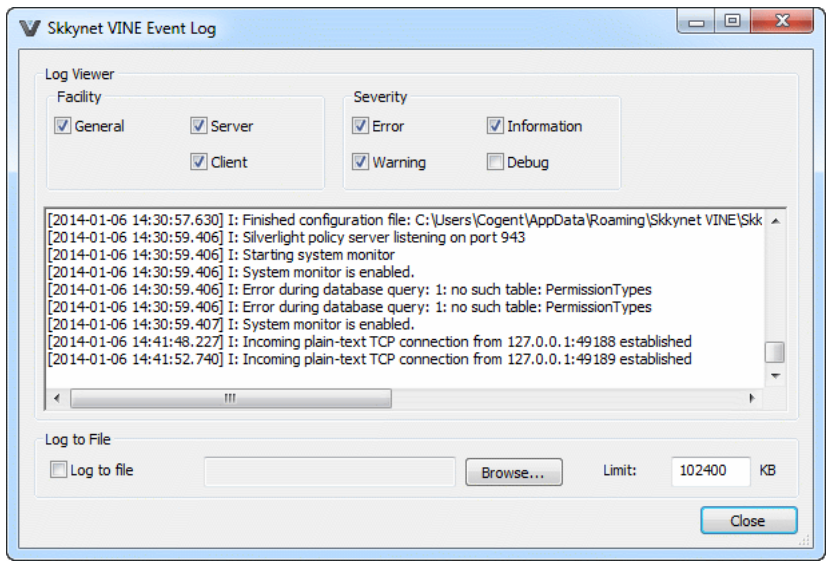
You have the following options for viewing the data:

- **Total** displays all connection statistics from the time that the client connection was first made.
- **Checkpoint** displays all connection statistics from the last time the Set Checkpoint button was clicked or from the time that the client started, whichever is later.
- **Performance** shows the performance of the connection, in terms of data changes per second, from the last time the Set Checkpoint button was clicked or from the time that the client started, whichever is later.
- **Set Checkpoint** records the time and current statistics for each client, for use by the Checkpoint and Performance options.

16.3. Event Log

This window lets you view a log of events from Skkynet VINE. You can open this window in either of two ways:

- Click the Event Log button in the Properties window.
- Right click the VINE icon in the system tray and select View Event Log from the pop-up menu.


Skkynet VINE Event Log
Log Viewer
Facility
General
Server
Client
Severity
Error
Information
Warning
Debug
[2014-01-06 14:30:57.630] I: Finished configuration file: C:\Users\Cogent\AppData\Roaming\Skkynet VINE\Skk
[2014-01-06 14:30:59.406] I: Silverlight policy server listening on port 943
[2014-01-06 14:30:59.406] I: Starting system monitor
[2014-01-06 14:30:59.406] I: System monitor is enabled.
[2014-01-06 14:30:59.406] I: Error during database query: 1: no such table: PermissionTypes
[2014-01-06 14:30:59.406] I: Error during database query: 1: no such table: PermissionTypes
[2014-01-06 14:30:59.407] I: System monitor is enabled.
[2014-01-06 14:41:48.227] I: Incoming plain-text TCP connection from 127.0.0.1:49188 established
[2014-01-06 14:41:52.740] I: Incoming plain-text TCP connection from 127.0.0.1:49189 established
Log to File
Log to file
Browse...
Limit:
102400
KB
Close

You can specify the category of event Facility and level of Severity by checking and unchecking the related items. It is possible to open multiple Event Log windows, and select different Facility and Severity options in each window.

The Debug option is very verbose. Operating in this mode could put an unusually high demand on system resources.

Log to File

If you wish to save a log, you can have Skkynet VINE write the log to a file. Just check the Log to File box and specify the file name in the entry field, typically with a `.log` extension. You can use the Browse button to find a file. The Limit text entry box allows you to change the maximum file size (default 102,400 Kb).

Once the `.log` file reaches the maximum size limit shown in the Limit field, it is renamed with a `.log.1` extension, and a new `.log` file is started. Combined, the two files provide at least as much data for analysis as specified in the Limit field. This also means that the actual size on disk of the two log files could be as much as twice the Limit size at any given time.

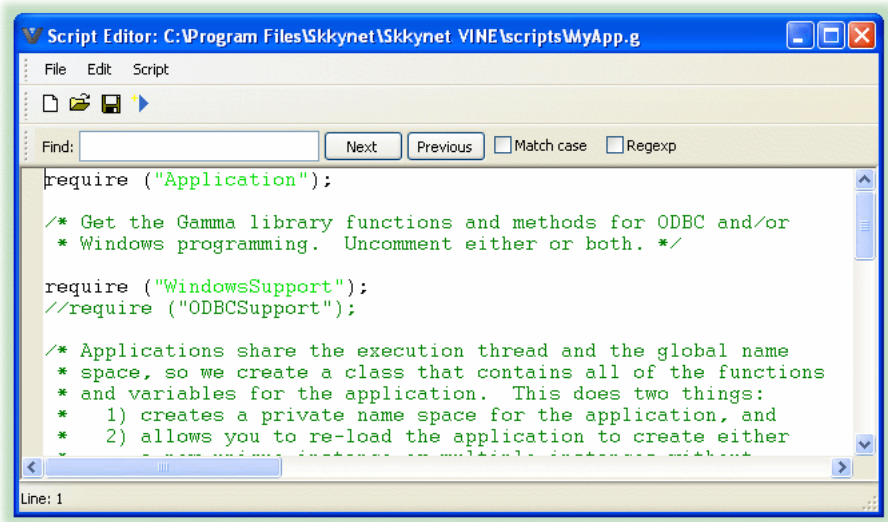
Here are some other useful pointers about log file management:

- If the file you specify doesn't exist, VINE will create it. If the file does exist, VINE will append log data to the file.
- As long as the Log to File box is checked, VINE will append all log entries to the file, even when restarted after a shutdown.

- Checking the Log to File button automatically causes the Event Log to log in the (verbose) Debug mode.

16.4. Script Editor

The Script Editor[1] lets you write and edit scripts.

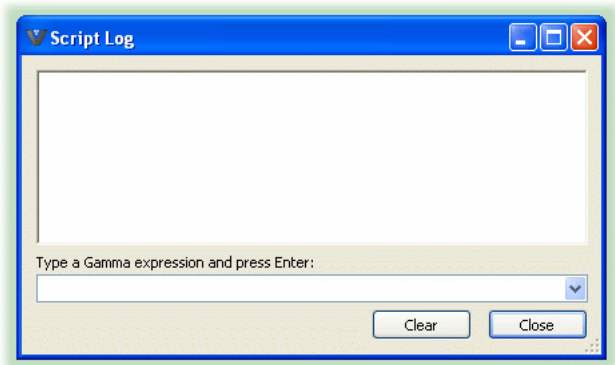

Script Editor: C:\Program Files\Skkynet\Skkynet VINE\scripts\MyApp.g
File
Edit
Script
Find:
Next
Previous
Match case
Regexp
require ("Application");
/* Get the Gamma library functions and methods for ODBC and/or
* Windows programming. Uncomment either or both. */
require ("WindowsSupport");
//require ("ODBCSupport");
/* Applications share the execution thread and the global name
* space, so we create a class that contains all of the functions
* and variables for the application. This does two things:
* 1) creates a private name space for the application, and
* 2) allows you to re-load the application to create either
Line: 1

The Script Editor offers features such as context-sensitive highlighting, prompted fill-ins for functions and variable names, automatic indenting, text string searches, and so on. Please refer to the VINE Scripting manual for detailed information.

16.5. Script Log

This window lets you view output from scripts, and interact with the Gamma scripting engine. You can open this window in either of two ways:

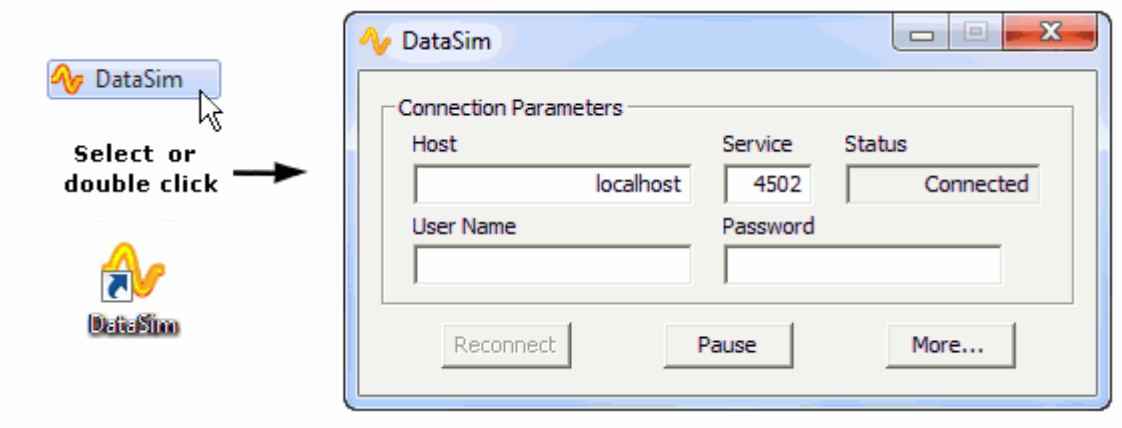
- Click the Script Log button in the Properties window.
- Right click the VINE icon in the system tray and select View Script Log from the pop-up menu.



Please refer to the VINE Scripting manual for detailed information about the Script Log.

16.6. DataSim - a data simulation program

DataSim is a data simulation program that creates local data for Skkynet VINE. It generates data for four different wave patterns, and sends these to VINE by a TCP connection.



As soon as DataSim starts, it attempts to connect to a VINE and begins generating data. To receive the data, Skkynet VINE should be set up as a tunnelling/mirroring master.

Connection Parameters

Host

The name or IP address of the host computer. Since DataSim connects via TCP, this can be any computer on the network running a Skkynet VINE acting as a tunnelling/mirroring master.

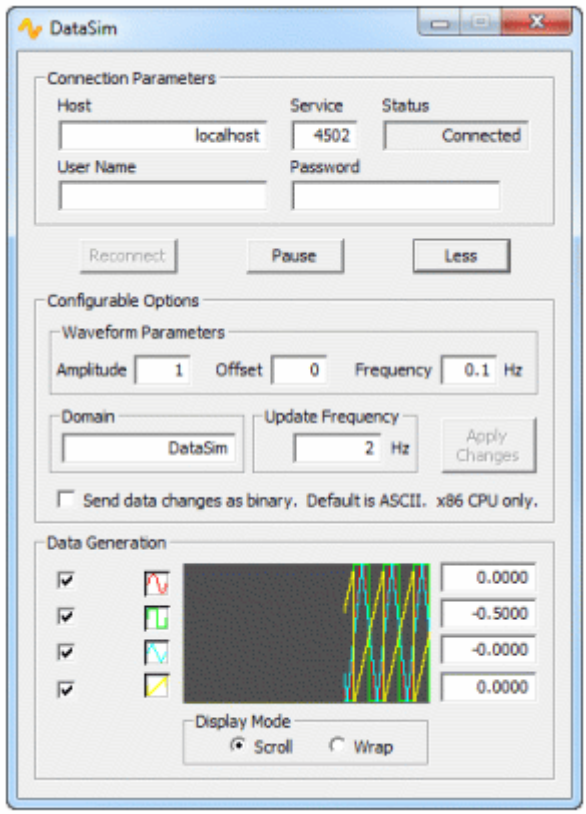
Service

The port number or service name as entered in the Master service/port entry box of VINE.

Status

Displays the attempts to connect, which change to Connected when the connection is made.

You can use the Reconnect button to reconnect and the Pause button to freeze all data generation. Press the Show button to view and change some of the data parameters:



Configurable Options

The following options can be set in DataSim, and sent to Skkynet VINE. All of the numeric options have a corresponding point in VINE that contains the value. Thus, these values can be set from within the Data Browser, by selecting the point name and entering a new value for it.

Waveform Parameters

Amplitude

The height of the wave forms. VINE Point name: `Amplitude`.

Offset

An offset from zero of the generated data, and thus the wave form. VINE Point name: `Offset`.

Frequency

The frequency of the wave form. VINE Point name: `Frequency`.

Data Domain

The Skkynet VINE data domain name for the data points. Any entry in this box changes the connection Status to `IDLE`. You must press the Reconnect button to reestablish the connection.

Update Frequency

The number of times per second that data changes and is sent to Skkynet VINE.

Apply Changes

Applies any changes that have been entered for these configurable options. This button is greyed out until a change has been entered and can be applied.

Send data change as binary...

Allows you to send the data changes from DataSim in binary form, rather than as ASCII characters. This can speed data update rates substantially. This feature is only available on `x86` machines.

Data Generation

For each of the following four variables, the check box stops or starts data generation, while the toggle button hides or shows the graph display. The numerical value is shown at the right of the graph.

SIN

Data generates a sine wave.

SQR

Data generates a square wave.

TRI

Data generates a triangular (45 degree) wave pattern.

RMP

Data generates a ramp wave—steadily increasing, followed by a sudden drop.

Display Mode

Controls how the data is displayed in the trend display. Scroll moves the wave from right to left as the data is generated. Old data scrolls off the left side of the display as new data scrolls in from the right. Wrap adds to the wave from left to right, and writes over old data.

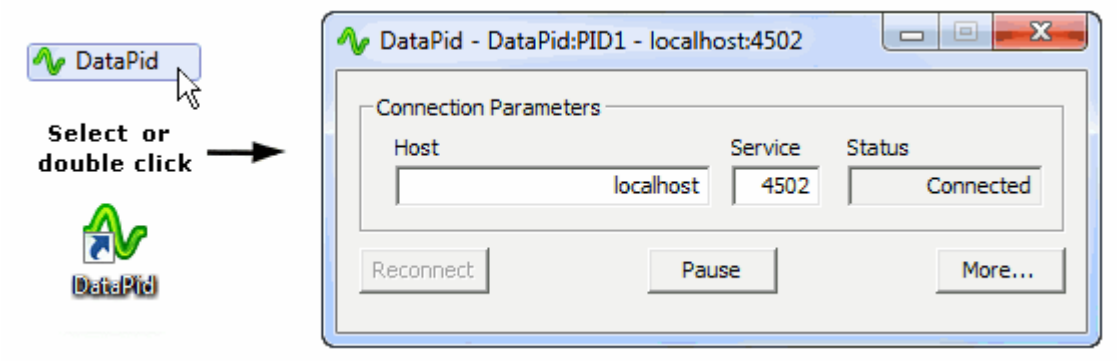
16.7. DataPid - a PID loop data simulation program

DataPid is a data simulation program for Skkynet VINE. It simulates data for a set point, control output, and process variable, and sends these values to VINE by a TCP connection.

This program is similar to DataSim, but with specialized data. A PID loop is often used by process control engineers to determine the efficiency of their system. A detailed explanation of PID loops is beyond the scope of this manual, but the data generated by this simulator can by used by anyone.

Starting Up

The DataPid can be started from the Windows Start menu, the command line, or by clicking on the desktop icon.



As soon as DataPid starts, it attempts to connect to a VINE and begins generating data. To receive the data, Skkynet VINE should be set up as a tunnelling/mirroring master.

The DataPid can take several command-line arguments at startup, to assist in running more than one instance at a time. Any combination of the following arguments can be supplied in the Target field of the shortcut, or on the command line.

/d *`data domain name`*

The name of the VINE data domain in which to write the data that DataPid generates. The default is `DataPid`.

/n *`pid name`*

The name of the PID loop. For example, if *`data domain name`* is `DataPid` and *`pid name`* is `PID1` then the data will the data will be created in a hierarchy beneath the point `DataPid:PID1`. The default is `PID1`.

/h *`host name`*

The name of the computer on which VINE is running. This can be an address or a name, for example, `127.0.0.1` or `developers.cogentrts.com`. The default is `localhost`.

/p *`port number`*

The port number on the target computer on which to connect. The default is `4502`.

/i

If specified, the DataPid window will be iconified when it starts.

Example:

```
DataPid.exe /i /d test /n pid
```

Connection Parameters

Host

The name or IP address of the host computer. Since DataSim connects via TCP, this can be any computer on the network running a Skkynet VINE acting as a tunnelling/mirroring master.

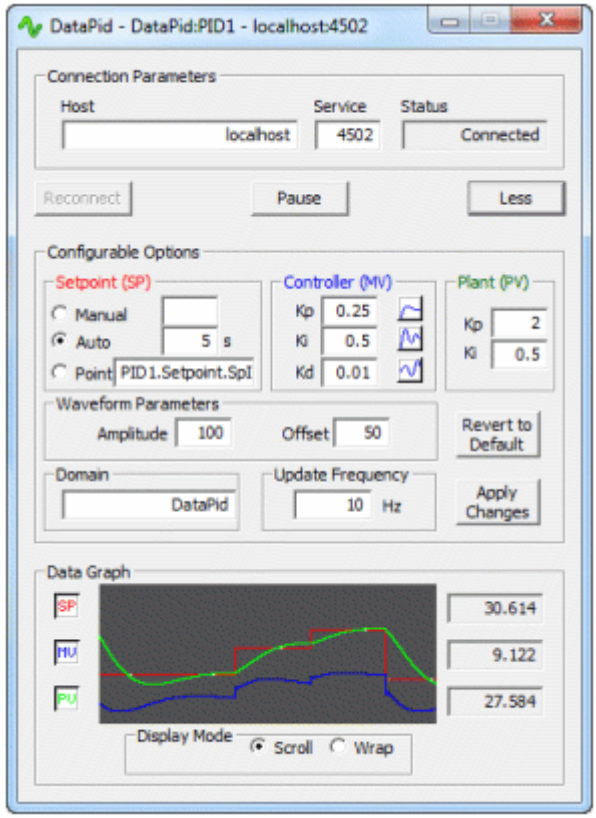
Service

The port number or service name as entered in the Master service/port entry box of VINE.

Status

Displays the attempts to connect, which change to Connected when the connection is made.

You can use the Reconnect button to reconnect and the Pause button to freeze all data generation. Press the More... button to view and change some of the data parameters:



Configurable Options

The following options can be set in DataPid, and sent to Skkynet VINE. All of the numeric options have a corresponding point in VINE that contains the value. Thus, these values can be set from within the Data Browser, by selecting the point name and entering a new value for it.

Setpoint (SP)

Manual	A number from `0` to `100` to for the set point. An entry here overrides Auto.
Auto	Changes the set point randomly, every *n* seconds, where *n* is the number entered.
Point	Changes the set point according to a point registered in VINE.

Controller (MV)

Kp	The proportional control factor, sets the speed of adjustment.	The wave form button sets a well-tuned PID loop.
Ki	The integral control factor, reduces error.	The wave form button sets a poorly-tuned PID loop.
Kd	The derivative control factor, provides a damping effect.	The wave form button sets an oscillating PID loop.

Plant (PV)

Kp	The proportional control factor of the plant.
Ki	The integral control factor of the plant.

Waveform Parameters

Amplitude	Not yet documented.
Offset	Not yet documented.

Data Domain

The Skkynet VINE data domain name for the data points. Any entry in this box changes the connection Status to `IDLE`. You must press the Reconnect button to reestablish the connection.

Update Frequency

The number of times per second that data changes and is sent to Skkynet VINE.

Apply Changes

Applies any changes that have been entered for these configurable options. This button is greyed out until a change has been entered and can be applied.

Data Graph

SP, MV, and PV

The toggle button hides or shows the graph display. The numerical value is shown at the right of the graph.

Display Mode

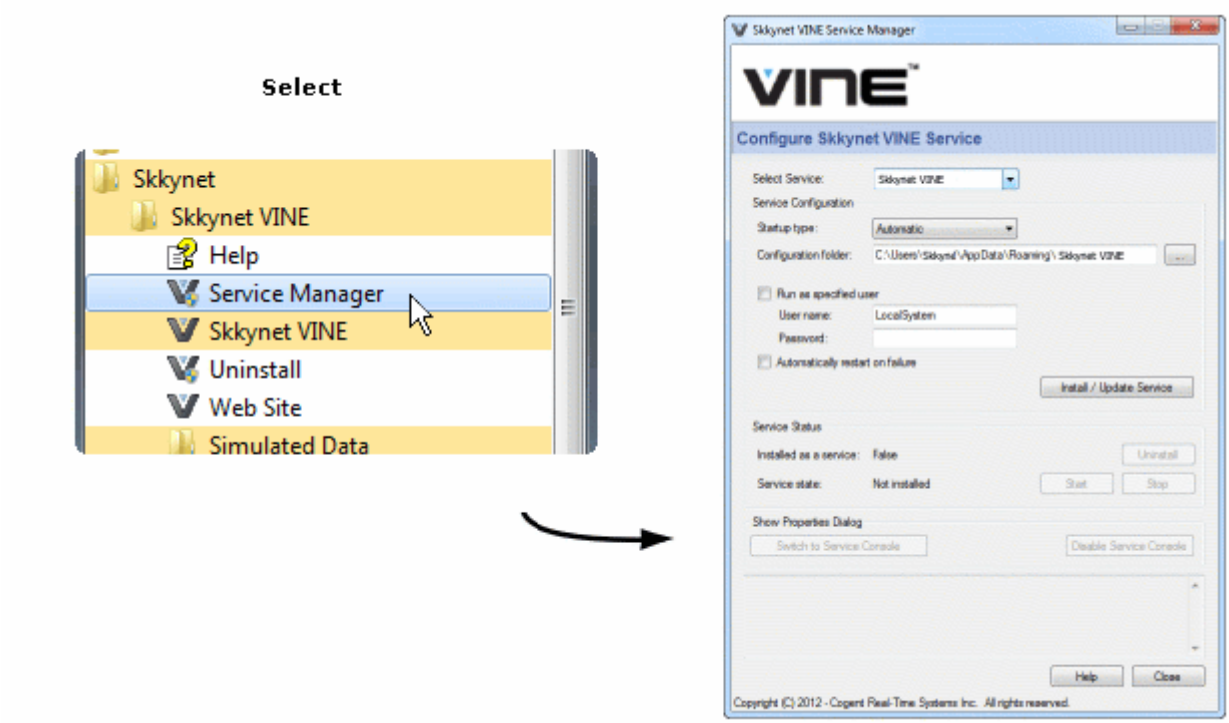
Controls how the data is displayed in the trend display. Scroll moves the wave from right to left as the data is generated. Old data scrolls off the left side of the display as new data scrolls in from the right. Wrap adds to the wave from left to right, and writes over old data.

16.8. Service Manager

The Service Manager is a program that provides access to Cogent software that can be installed to run as a Windows service. With this program you can select and configure how the service runs, change its status, and open the Properties window of Skkynet VINE.

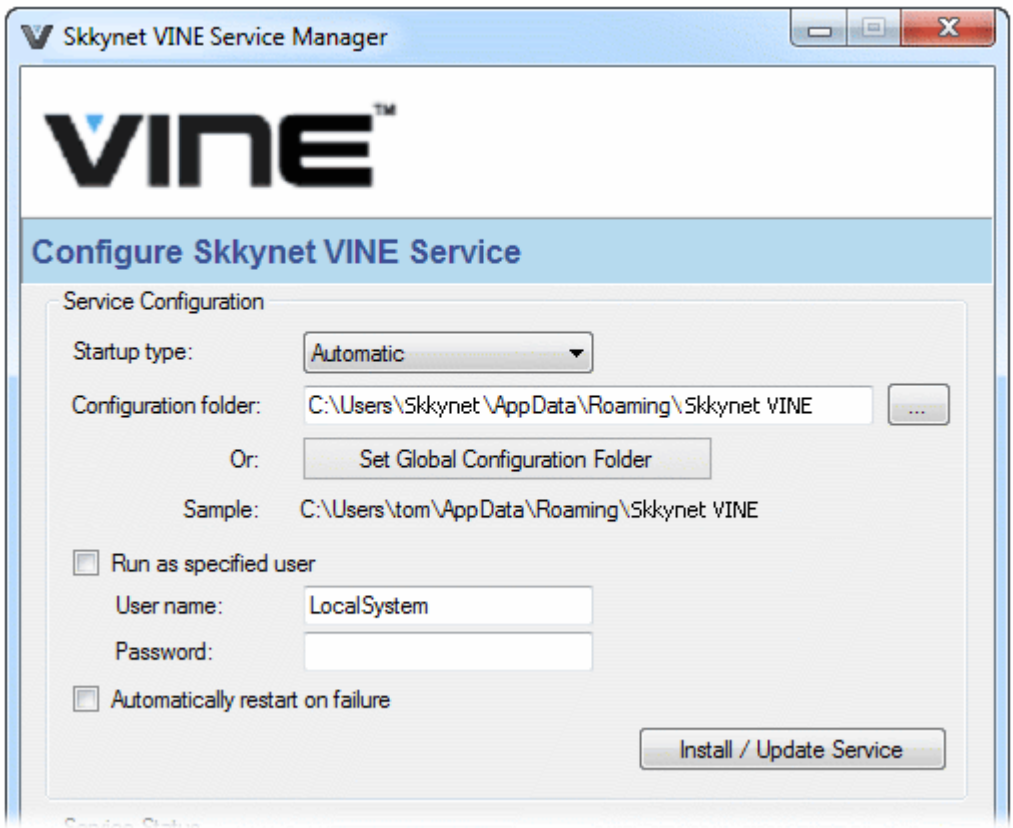
Starting the Service Manager

The Service Manager can be started from the Skkynet VINE program group in the Cogent entry of the Windows Start menu.



Once started, you can choose the service you need to configure from the Select Service dropdown list at the top. Then you can configure, install, and check the status of the service, as well as view the VINE Properties window. The scrolling list at the bottom maintains a record of activities.

Service Configuration



Startup type:

Choose between Automatic, Manual, or Disabled to specify how you want the service to start when Windows starts.

Configuration folder:

Allows you to specify a folder in which to put the configuration files for Skkynet VINE. Typically this does not need to be changed. Please refer to Section 1.5, *Configuration Files* for more information about configuration files.

The Set Global Configuration Folder button lets you specify the configuration file through the registry. This will preserve the configuration directory even after uninstalling as a service.

Run as specified user

It is recommended that you run VINE as the local SYSTEM (`LocalSystem`) user, and leave this box unchecked.

The VINE Properties window is only available when running the service as the local SYSTEM user. If you run as a specified user then you will not be able to access the Properties window to make changes to the VINE while it is running as a service.

However, there are certain situations in which you may need to run it as a specified user. To specify a user other than the local SYSTEM user, enter the User name and Password as applicable, then please see Appendix E, *Running as a Windows Service (Specified User)* for important additional information

Automatically restart on failure

Have the service restart should it fail or be stopped for some reason.

Install/Update Service

When the above configuration is complete, press this button to install VINE as a service. This will also start the VINE service, though on some systems the service may need to be started manually if it doesn't start with the install operation. If the service is running and you make changes to the configuration, pressing this button will cycle through a service shutdown and restart to apply your changes.

Service Status

Installed as a service:

Indicates whether the selected program is installed as a service or not (True or False). The Uninstall button allows you to uninstall VINE as a service.

Service state:

Indicates the run status of the service (Stopped, StartPending, Running, etc.) . The Start and Stop buttons allow you to start or stop the service.

Show Properties Dialog

This option appears differently for different versions of the Windows operating system:

Windows XP and Windows Server 2003

A Show Properties Dialog button allows you to open the Skkynet VINE Properties window.

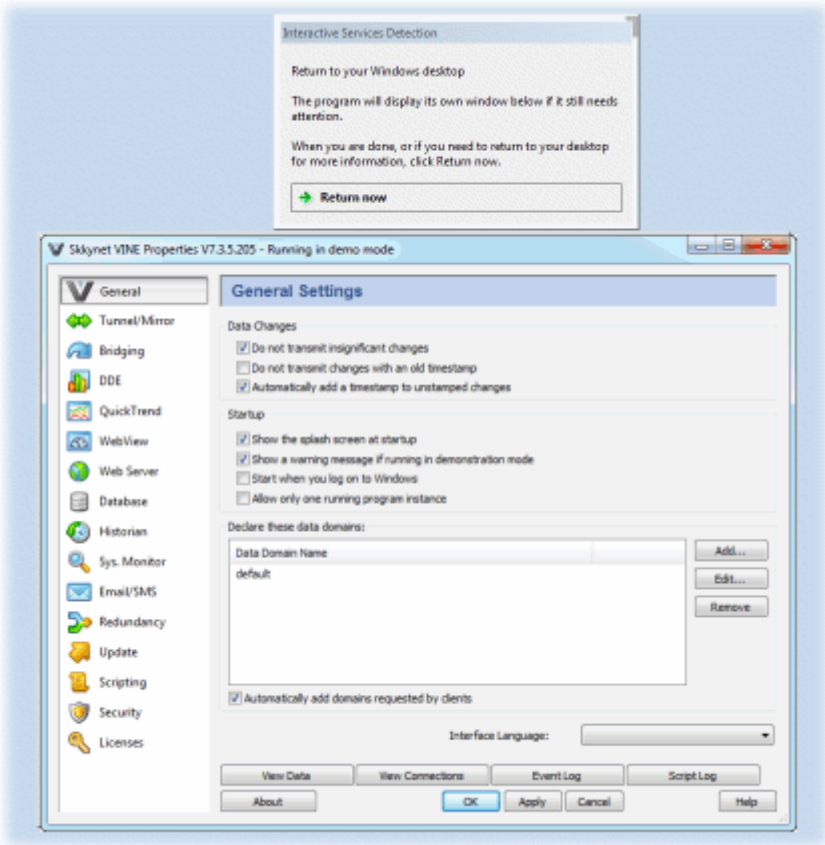
The VINE Properties window is only visible on the primary console of the computer running Skkynet VINE. If you are currently logged in via a remote desktop session, you will see a pop-up dialog indicating that you must be connected to the computer's primary console. You can do this by using the `/admin` or `/console` options on the Microsoft Remote Desktop client.

If you are already connected to the primary console, the VINE Properties window and system tray icon will be displayed when you press Show Properties Dialog button.

Windows Vista, Windows 7, Windows 8, Windows Server 2008, and Windows Server 2012

The Switch to Service Console button displays the Windows Service Console, which is where the VINE Properties window appears when running as a service. This allows you to view data and make changes to the DataHub configuration while it is running as a service.

The Service Console is a special display console provided by Microsoft Windows, also known as the "session 0 console". This was introduced in Windows Vista as a security mechanism to limit access to the user interface of high-permission processes. When you switch to the Service Console, your desktop will be hidden and the screen background will change color to indicate the special status of the Service Console.

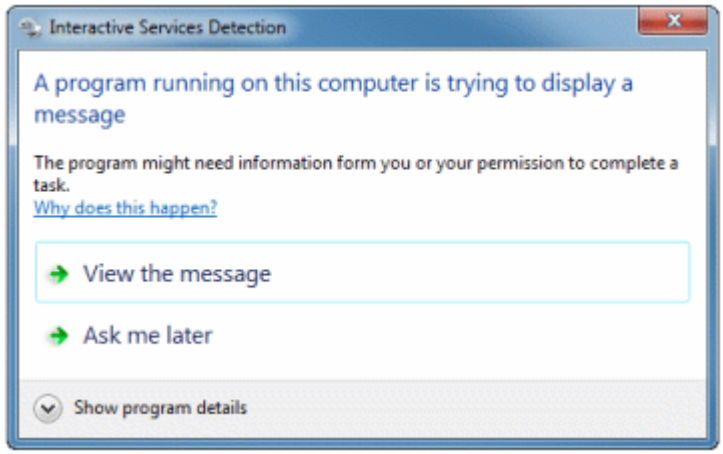


While the Service Console is open, your other applications will continue to run normally, but you will not be able to see or interact with them. A dialog box will be visible while you are viewing the Service Console that will allow you to switch back to your regular desktop at any time. If you have other system services running on your computer that also have user interface windows, those services will also be available to you while viewing the Service Console.

When you have finished viewing or editing the VINE properties, before returning to the normal user console, you should click the Apply and OK buttons to close down the VINE Properties window. Also be sure to close any other VINE windows. This prevents the Windows Interactive Service Detection program from displaying pop-up messages when you return to the normal user console.

Once all the VINE windows are closed, you can return to the normal user console by pressing the Return Now button in the Interactive Service Detection window.

If you forget to close any of the VINE windows while working in the Service Console, Windows will begin popping up messages in the user console telling you there is a program requiring attention in the Service Console.


Interactive Services Detection
A program running on this computer is trying to display a message
The program might need information form you or your permission to complete a task.
Why does this happen?
View the message
Ask me later
Show program details

You can stop these messages and close down the Service Console by clicking on the Disable Service Console button in the VINE Service Manager. Clicking this button will not stop the VINE service.
When you close the Service Manager it will automatically disable the service console. This will stop Windows from periodically displaying the Interactive Services Detection dialog on your system.

Notes

1. This editor is based on the Scintilla and SciTE editor.

Chapter 17. Using VINE Commands

VINE has an internal command set, documented in the Reference I, *Skkynet VINE Command Set* reference. When you change the configuration of VINE in the Properties window, one or more of these commands is written in the configuration file, and VINE receives that command every time it starts up. You can use these commands to create custom configuration files. (Please see Section 1.5, *Configuration Files*.)

It is also possible to issue these commands to VINE during run-time in any of the following ways:

- With a VINE script, using the special `datahub_command` function.
- Using the Skkynet VINE APIs for C++, Java, and .NET.
- Over a direct TCP connection.

This is how custom applications can interact directly with VINE. For example, DataSim connects to VINE by using the Skkynet VINE APIs for C++, Java, and .NET.

17.1. Command Syntax

VINE commands have the following syntax:

(*command* arg_1 arg_2 ...)

The whole command must be surrounded by parentheses. The command name and its arguments are each separated by white space—single spaces, tabs, or carriage returns are allowed. For example, the following line of a custom configuration file tells Skkynet VINE to create a new data domain, named `TestDomain`.

```
(create_domain TestDomain)
```

Multiple-word strings must be in quotes. Numbers take their own values. Booleans are `0` for false and `1` for true.

17.2. Return Syntax

When Skkynet VINE executes a command, it may return a success message or an error message. The returned message contains the original command, and some or all of the arguments for the command. These messages will be received by programs using the Skkynet VINE APIs for C++, Java, and .NET, but they will not be received by VINE scripts. The two types of success messages are:

- **No arguments:** (success *command*)
- **One or more arguments:** (success *command* arg_1)

Success messages are returned if the VINE command **acksuccess** has been previously issued with a value of `1`. A value of `0` means no success messages will be returned. Error messages are returned any time there is an error. There are four types of error messages:

- **No arguments:** (error "-2: (*command*): *error message*")
- **One argument:** (error "-2: (*command* arg_1): *error message*")
- **Two arguments:** (error "-2: (*command* arg_1 arg_2): *error message*")
- **More than two arguments:** (error "-2: (*command* arg_1 ...): *error message*")

The error messages are the negation of these error codes:

`ST_OK`	The function executed without error.
`ST_ERROR`	An error occurred.
`ST_NO_TASK`	A required task does not exist.
`ST_NO_MSG`	There is no message available.
`ST_WOULDBLOCK`	This action would block, and is not permitted.
`ST_INTR`	An interrupt occurred.
`ST_FULL`	The queue is full.
`ST_LOCKED`	A VINE point is locked.
`ST_SECURITY`	The security level is insufficient.
`ST_NO_POINT`	A required VINE point does not exist.
`ST_INSIG`	A change in a VINE point's value is insignificant. This is not really an error, but a notification that no exception will be generated by the datahub.
`ST_UNKNOWN`	There is an unknown error.
`ST_NO_QUEUE`	A target task has no queue, or **qserve** is absent.
`ST_CMD_SYNTAX_ERROR`	The command was not found, or there was a syntax error.

`ST_REPLIED`	The reply was complete.
`ST_WRONG_TYPE`	The type of a point or variable was wrong.
`ST_TOO_LARGE`	A value to be written to memory is larger than the available buffer.
`ST_NO_MEMORY`	There is insufficient memory available.
`ST_OLD_DATA`	Time-significant data is out of date.
`ST_TIMEOUT`	A timeout occurred in poll mode.

17.3. Sending Commands by TCP

It is possible to send commands directly to VINE over TCP, by using a tunnelling/mirroring connection. The communication between a client and VINE over TCP follows these guidelines:

- The connection is a single bi-directional socket.
- All communication from VINE to the client is non-blocking and asynchronous. It is possible to use blocking I/O in the client, and to wait for a response from VINE as well, but we don't advise doing either.
- Since TCP is streamed, there are no packets as such. Each message starts with an open parenthesis and ends with a matching closing parenthesis. Messages should be terminated with a newline (`\n`) character. If they are not, then VINE will hold off on processing the incoming messages until it receives a newline character, and then it processes all messages in order, in a batch. There is a maximum character length allowed (around 1 MB) for a batch of messages, after which VINE will discard data until it sees a newline.
- Parameters within a message can themselves be parenthesized expressions. For example, the following message contains a command and five parameters:

```
(OPCAddItem server1 item1 0 default:server1.item1
 (default server1 item1))
```

  The fifth parameter is itself a command: `(default server1 item1)`.

- All strings in a message are surrounded by double quotes. Inside the double quotes, the sequence `\"` embeds a double quote, `\\` embeds a `\` character, `\n` embeds a newline, `\t` embeds a tab, `\f` embeds a form feed, `\r` embeds a carriage return. `\` followed by any other character produces that character with the `\` removed. Parentheses inside double quotes do not match parentheses outside double quotes.
- You can embed any character inside a non-quoted string by putting a `\` in front of that character, so the string `abc\ def` would be the same as `"abc def"`.
- Example of using a TCP socket directly without going through the Cogent C API:

```
SOCKET s;
char buf[256];
int len;
char *pointname = "testpoint";
double value = 1.0;

len = sprintf(buf, "(cset \"default:%s\" %g)\n", pointname, value);
send(s, buf, len, 0);
```

There is a function in the C++ header file that parses this kind of stream, called `UT_LispParse`, and another function called `UT_LispString` that can help a bit with writing lisp expressions. They automatically add double-quotes around `%s` formatted strings, and escape characters within the string.

Chapter 18. Troubleshooting

This chapter has answers to the most common troubleshooting questions that may arise. Any other known, unresolved issues are discussed in the `README.txt` which is included in the distribution archive. Please read that file and this page before contacting Cogent for help.

For responses to other common questions and issues, please see also the Cogent Community Forum Frequently Asked Questions.

1. I can't get a connection.

Check the following:

- Is your connection to the Internet and/or network up and functioning normally?
- Does your firewall allow a local program (like Skkynet VINE) to act as a server?
- Does Skkynet VINE run OK?
- Is VINE installed and configured correctly?
- For drag and drop operations, is the Drag & Drop Style at the bottom of the Data Browser set to MS-Office (Excel/Word)?
- Are you using the VINE Web Server and is Skype installed on the same machine? There is a default option in Skype (follow the Tools - Options - Connection menus) to use port `80` as an alternative for incoming connections. You will need to either disable this, or change the port number of VINE's web server to something other than `80`.

2. Windows Error Messages

For a list of Windows error messages, including those for TCP and DDE, please refer to Appendix G, *Error Messages*.

3. Skkynet VINE Error Messages

Here is a brief explanation of some of the more common error messages you might encounter when using Skkynet VINE:

License failure: License in use by master

There are two possible reasons for this message:

1. One VINE is tunnelling/mirroring to another VINE on another computer and that VINE is tunnelling/mirroring back to the first.
2. VINE is tunnelling/mirroring to itself.

Either of these are probably due to misconfiguration. Check your configuration files to ensure that only one VINE is configured to be a master in any master/slave pair, and that no VINE is tunnelling/mirroring to itself.

4. Messages from Excel

When you save and close a spreadsheet connected to VINE and then attempt to reopen it, you may get one or more messages, depending on your security settings in Excel, or other circumstances. Here's a summary of each message, and what to do:

This document contains macros. Enable them?

Click Enable Macros.

This workbook contains links. Update them?

> Click Update. If the VINE is already running, all the links should then update automatically. If not, you may get a #REF! entry in some cells, and the next message (see below) will probably appear.

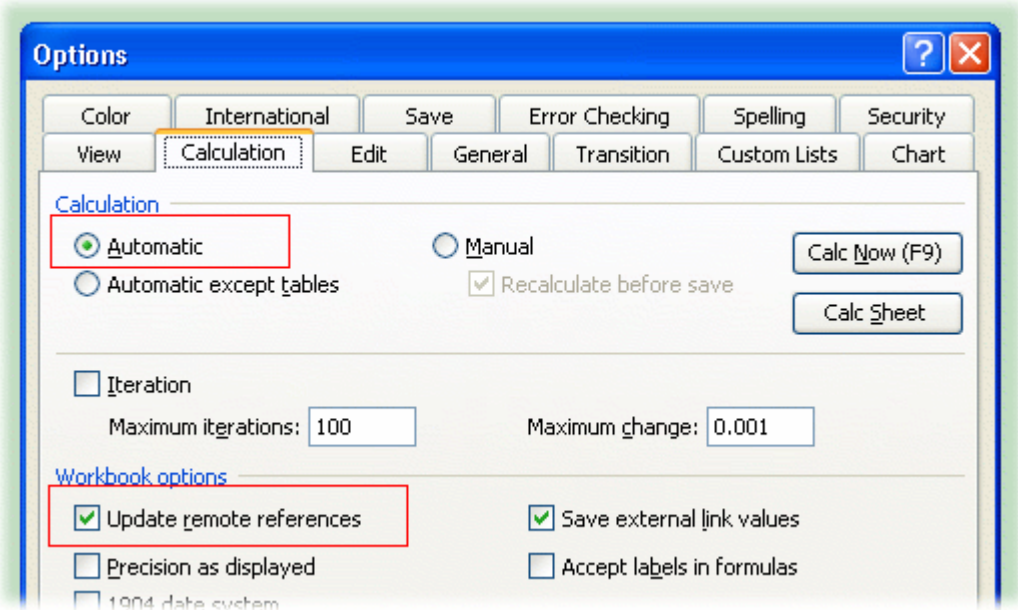
Remote data not accessible. Start VINE?

> Click No. At this point the best thing to do is close the worksheet, start the requested program, and then reopen the worksheet. When you update the spreadsheet (see above) this time you won't get any #REF! entries.

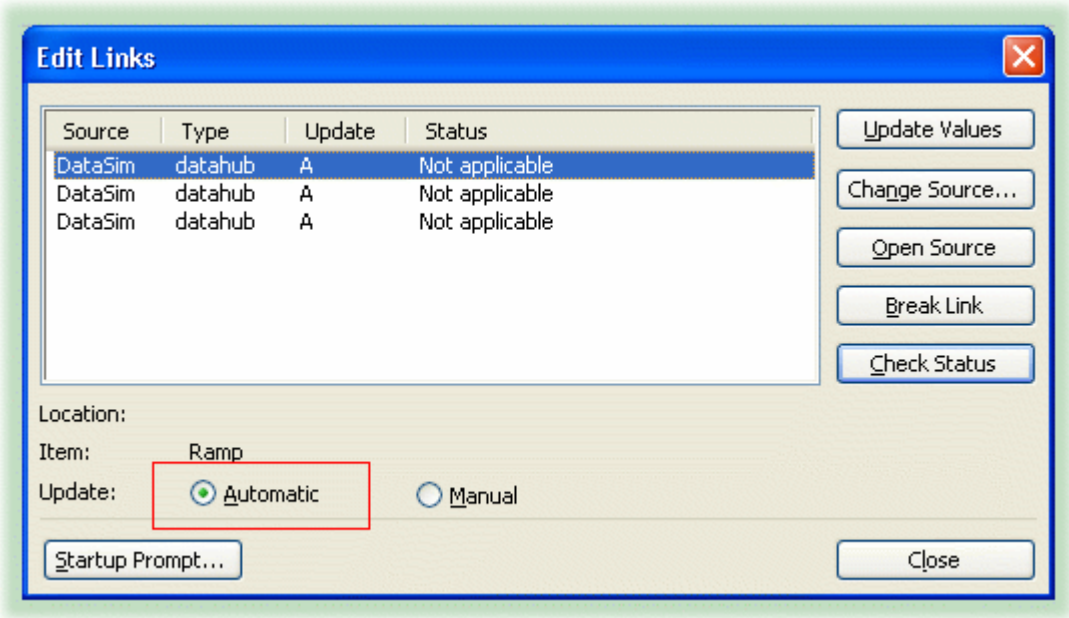
5. Excel not updating

The default settings in Excel allow you to drag and drop from Skkynet VINE into your spreadsheet and see the data updating automatically. Sometimes however the Excel configuration may have been changed so that you do not see this. For example, if you drag a data point into Excel and you get the first value, but then nothing after that, you may want to check the following settings.

1. From the Tools menu, choose Options to open the Options window.



2. Ensure that the Automatic option in Calculation is selected.
3. Ensure that the Update remote references option in Workbook options is selected. Then close the Options window.
4. From the Edit menu, choose Links to open the Edit Links window.



5. Ensure that the Automatic option for Update is selected. Then close the Edit Links window.

For responses to other common questions and issues, please see also the Cogent Community Forum Frequently Asked Questions.

Appendix A. DDE Overview

DDE (Dynamic Data Exchange) is a well-established mechanism for exchanging data among processes in MS-Windows. The mechanism was intentionally designed to be easy to use and to represent data as simply as possible. DDE is implemented in many popular programs that run in Windows, such as Microsoft Excel and Microsoft Word. This widespread availability makes DDE a good choice for general data sharing.

The competition with DDE is COM, with its variants for OLE: OPC and ActiveX. By comparison, DDE is simpler, and therefore faster, than the equivalent COM interface if implemented as a separate process. DDE is much easier to implement in code, and offers a particular data model as (name, value) pairs. In the case of real-time data, this model is well suited, and therefore offers the best cost/benefit ratio when programming for real-time data.

However, DDE was not designed to be used over a network. The Cogent solution for this shortcoming is to tunnel/mirror two copies of Skkynet VINE over a network or the Internet using TCP. Thus, two programs that use only DDE can exchange data across a robust, TCP-enabled link.

Data Definitions

DDE defines data in terms of (service, topic, item), explained as follows:

service

A name used by a DDE server to identify its service to DDE clients. The default DDE service name for Skkynet VINE is `vine`. Unlike most Windows programs, Skkynet VINE lets you change this name, or add more names if you'd like.

topic

A way to categorize items. This corresponds to a Skkynet VINE *data domain*.

item

A variable that holds a value. This corresponds to a Skkynet VINE *point*.

Here are some service and topic names for several Windows programs:

Application	DDE Service Name	DDE Topic Name
Skkynet VINE	Multiple names can be assigned. The default name is `vine`.	Any Skkynet VINE domain name, the default domain is `default`.
Microsoft Excel	`EXCEL`	The name of the spreadsheet, chart, macro, etc. For example: `mysheet.xls`
Microsoft Access	`MSACCESS`	The name of the table, SQL query, or macro to run.
Microsoft Word	`WINWORD`	The name of the document, including the .doc extension.
FoxPro	`FOXPRO`	
InTouch Viewer	`VIEW`	`TAGNAME`
FIX DMACS	`DMDDE`	`DATA`
National Instruments' Lookout	`LOOKOUT`	The name of the application, without the `.lkp` extension.

Application	DDE Service Name	DDE Topic Name
Asymetrix Toolbook	`TOOLBOOK`	The name of the toolbook application, with the `.tbk` extension.

Here are a few service and topic names for financial data feeds:

Data Feed	DDE Service Name	DDE Topic Name	Symbol Example
ADP Shark	`ML`	`LP`	`IBM.N.Q`
Bloomberg	`BLP`	`M`	`IBM EQUITY,[LAST TRADE]`
Bridge	`BDDE`	`TKR`	`@USH8/LS`
FXCM	`FXPS`	`BID`	`EUR/USD`
Future Source (CalcSource)	`CALCSRC`	`P`	`USH8.LAST`
Future Source (ProfNet)	`PROFDDE`	`LIVE`	`APIU02,LAST`
Knight Rider Profit Center	`QMASTER`	`QUOTE`	`USH8.LAST`
METATRADER	`MT`	`BID`	`USDCHF`
METATRADER v. 4	`MT4`	`BID`	`USDCHF`
MGFOREX	`MGFOREX`	`RATES`	`USD`
Moneycast	`WBSERVER`	`SES`	`19/L (19 is a stock code)`
Reuters	`REUTER`	`IDN`	`USH8 /IBM ,LAST`
Telerate WorkStation	`TWINDDE`	`QUOTES`	`USH8.3 LAST`
Universal Market Data Server	`USDDE`	`QUOTE`	`US8H;LAST`

Client and Server

In DDE, the role of client and server in data exchange is fairly clear. A client initiates the activity, and the server responds. To facilitate two-way data transfer, each Skkynet VINE can each act as both client and server. This is what allows Excel spreadsheets to share data bidirectionally across a network.

Sending and Receiving Data

There are three ways to send and receive ordinary data over DDE. Here is a brief explanation, using Skkynet VINE and Microsoft Excel as examples:

Poke The client sends data for an item directly to the server. In Excel, this is done with a macro. The server does not necessarily reply. The actual data flow is from client to server—from Excel to VINE.

Request The client asks the server to send an item's data. In Excel, this is done with a macro. The client receives either the requested value, or `NULL` if the server can't send the value. The actual data flow is from server to client—from VINE to Excel.

Advise The client asks to be notified of any change in the data for an item. If the server agrees to the request, it sends the new value for the item each time its value changes. Skkynet VINE can conduct two-way communication with Excel using only this advise capability, as follows:

- To receive data into Excel from VINE, you set VINE to act as a DDE server, which requires you to enter a service name. This identifies VINE to Excel. Then you can drag and drop a point name from VINE into a cell of an Excel spreadsheet.

- To send data from Excel to VINE, you set VINE to act as a DDE client, which requires you to enter service, topic, and item names. These identify the item in the spreadsheet to VINE. Then, in Excel, you name a cell with the VINE point name.

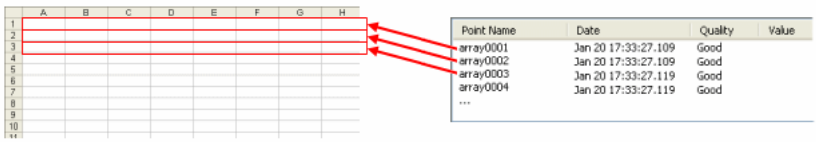
Appendix B. Excel Macro Library

This set of Excel macros each work on a 100 row x 40 column table of data in `Sheet1` of the worksheet, starting at cell position `A1`. We have tried to make these macros generic so you can easily modify them to suit your needs.

B.1. Configure Excel to receive data from Skkynet VINE (using DDEAdvise)

These macros normally need to be run only once, when first setting up a spreadsheet to receive data.

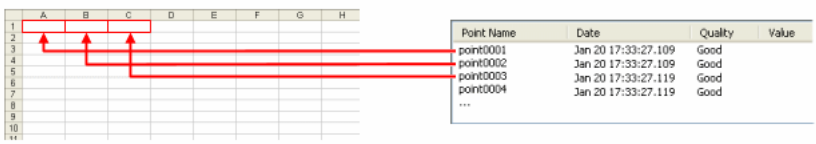
- **Attach array data in Skkynet VINE, one array per row, to a table of values in Excel.** It is often more convenient to transmit large sets of Excel data as an array because this significantly reduces the bandwidth requirements and increases the speed of transmission. This macro sets up **DDEAdvise** loops from VINE to Excel, so that each row of the table is linked to an array point in VINE.



Each data point represents a row of data. This macro assumes the names are "array0001", "array0002", etc.

```
-------------------------------------------------
Sub register_arrays()
    Dim pname As String

    For i = 1 To 100
        pname = Format(i, "0000")
        pname = "=vine|default!array" & pname
        Worksheets("Sheet1").Range(Cells(i, 1), Cells(i, 40)).FormulaArray = pname
    Next i
End Sub
-------------------------------------------------
```

- **Attach individual point data in Skkynet VINE, one point per cell, to a table of values.** This macro sets up **DDEAdvise** loops from VINE to Excel, so that each cell in the table is linked to a point in Skkynet VINE.



This macro assumes that the data points are named "point0001", "point0002", etc.

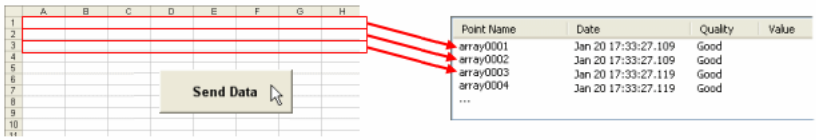
```
-------------------------------------------------
Sub register_points()
    Dim pname As String

    For i = 1 To 100
        For j = 1 To 40
            pname = Format((i - 1) * 40 + j, "0000")
            pname = "=vine|default!point" & pname
            Worksheets("Sheet1").Cells(i, j).Formula = pname
        Next j
    Next i
End Sub
```

```
-------------------------------------------------
```

B.2. Write data from Excel - User initiated (using DDEPoke)

These macros are useful for writing data out from Excel on demand. In other words, the user decides when to write the data, and does so by running one of these macros (usually from an assigned button click).

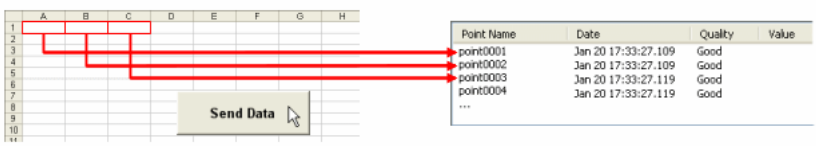
- **Transmit array data, one array per row, to points in Skkynet VINE.** Triggering this macro writes all the data from the table in Excel to Skkynet VINE. The macro writes each row of the table as an array point in VINE. All rows of the table get transmitted, one after another.



This macro assumes that the data points are named "array0001", "array0002", etc.

```
-------------------------------------------------
Sub transmit_arrays()
    Dim chan As Integer
    Dim pname As String

    chan = DDEInitiate("vine", "default")
    For i = 1 To 100
        pname = Format(i, "0000")
        pname = "array" & pname
        DDEPoke chan, pname, Worksheets("Sheet1").Range(Cells(i, 1), Cells(i, 40))
    Next i
    DDETerminate (chan)
End Sub
-------------------------------------------------
```

- **Transmit individual point data, one point per cell, to points in Skkynet VINE.** Triggering this macro writes all the data from the Excel table to VINE. The macro writes each cell of the table in turn to a single point in VINE.



This macro assumes that the data points are named "point0001", "point0002", etc.

```
-------------------------------------------------
Sub transmit_points()
    Dim chan As Integer
    Dim pname As String

    chan = DDEInitiate("vine", "default")
    For i = 1 To 100
        For j = 1 To 40
            pname = Format((i - 1) * 40 + j, "0000")
            pname = "point" & pname
            DDEPoke chan, pname, Worksheets("Sheet1").Cells(i, j)
        Next j
```

```
    Next i
    DDETerminate (chan)
End Sub
-------------------------------------------------
```

B.3. Write data from Excel - Automatically on value change (using DDEPoke)

These macros are useful for automatically transmitting data from Excel into Skkynet VINE.

- **Emit new cell values to Skkynet VINE.** Whenever a user enters a new value, this macro checks to see if that cell is named. If so, the macro emits the new value to a Skkynet VINE point of the same name. The subroutine name "Worksheet_Change" is special - it is called by Excel whenever a change occurs on the Worksheet due to user input or recalculation (though not a change due to a DDE message; for that see Other Useful Macros).

  ```
  -------------------------------------------------
  Sub Worksheet_Change(ByVal Target As Range)
      Dim rname As String
      Dim channel As Variant

      On Error Resume Next
      rname = Target.name.name
      If Not rname = "" Then
          channel = DDEInitiate("vine", "default")
          DDEPoke channel, rname, Target
          DDETerminate (channel)
      End If
  End Sub
  -------------------------------------------------
  ```

- **Transmit changes to a range.** This pair of macros determines that a cell within a particular named range has changed through user input, and transmits the contents of the range to Skkynet VINE. This is useful because you do not have to configure each cell you want to write out to VINE. If the cell that is changed lies within a defined range, then all values in that range are automatically written out to VINE.

  The `Worksheet_Change` routine determines the enclosing range for the change, and if the range matches one of a predefined set, it will send that range to Skkynet VINE. The `NameOfParentRange` function determines the name of the cell range that intersects a given range. If more than one named range in the worksheet intersects the given range, it returns only the first one.

  Add to Workbook Macro Code:

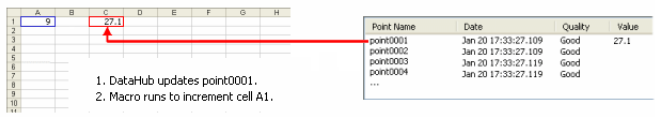
  ```
  -------------------------------------------------
  Function NameOfParentRange(Rng As Range) As String
      Dim Nm As Name
      For Each Nm In ThisWorkbook.Names
          If Rng.Parent.Name = Nm.RefersToRange.Parent.Name Then
              If Not Application.Intersect(Rng, Nm.RefersToRange) Is Nothing Then
                  NameOfParentRange = Nm.Name
                  Exit Function
              End If
          End If
      Next Nm
      NameOfParentRange = ""
  End Function
  -------------------------------------------------
  ```

Add to Sheet1 macro code:

```
------------------------------------------------
Sub Worksheet_Change(ByVal r As Range)
    Dim pname As String
    Dim chan As Integer

    pname = ThisWorkbook.NameOfParentRange(r)
    If Not pname = "" Then
        On Error Resume Next
        chan = DDEInitiate("vine", "default")
        DDEPoke chan, pname, Worksheets("Sheet1").Range(pname)
        DDETerminate (chan)
    End If
End Sub
------------------------------------------------
```

B.4. Other Useful Macros

- **Cause a macro to run when a Skkynet VINE point changes value.** Here is one macro that runs another macro every time a certain cell's value is updated by a DDE message. The macro that gets run is `link_updated`. It simply increments the value in cell `A1`. You can easily change this example to meet your needs. The `set_link` macro tells the workbook to run the `link_updated` macro whenever VINE sends a DDE message about `point0001`. You can also change the name of the VINE point as needed.



Add to Sheet1 macro code:

```
------------------------------------------------
Sub link_updated()
    Cells(1, 1) = Cells(1, 1) + 1
End Sub
------------------------------------------------
```

Run once to establish the link:

```
------------------------------------------------
Sub set_link()
    ThisWorkbook.SetLinkOnData "vine|default!'point0001'", "Sheet1.link_updated"
End Sub
------------------------------------------------
```

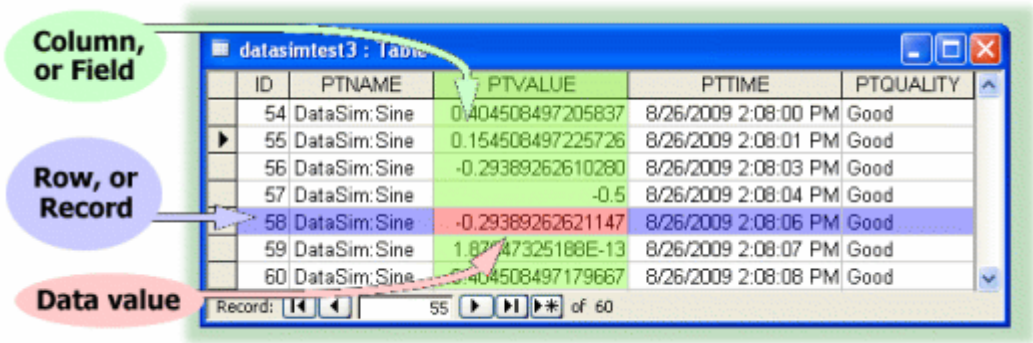
Appendix C. ODBC Database Concepts

Database Terminology and Concepts

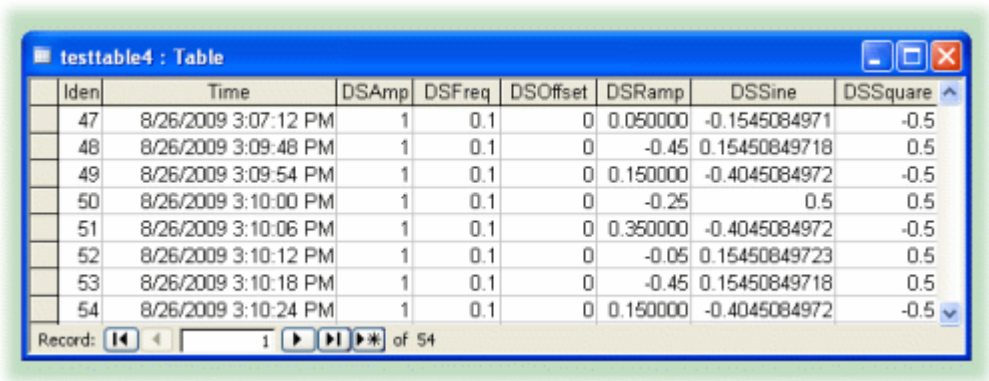
In general, a *database* is a collection of information. Computerized databases store data in *tables*, which are accessed through a *database management system* or *DBMS*, such as SQL Server, MS Access, MySQL, Oracle, etc. All of these are capable of storing data in related, linked tables, which taken together are called a *relational database*. Most modern computerized databases are relational databases.

A database *table* is a logical grouping of related data, organized by the common attributes or characteristics of individual data items. Each *column* or *field* in the table contains a particular attribute, and is of one particular *data type*. Typical data types include boolean, string, numeric, date/time, etc. Each *row* or *record* in the table contains a complete set of every *data value* related to a single item.

For example, a table containing data from Skkynet VINE might have columns (fields) for a point name, value, timestamp, and quality. Each row (record) would show the various data values logged for that point at different times:


Column, or Field
Row, or Record
Data value
datasimtest3 : Table
ID
PTNAME
PTVALUE
PTTIME
PTQUALITY
54 DataSim:Sine 0.404508497205837 8/26/2009 2:08:00 PM Good
55 DataSim:Sine 0.154508497225726 8/26/2009 2:08:01 PM Good
56 DataSim:Sine -0.29389262610280 8/26/2009 2:08:03 PM Good
57 DataSim:Sine -0.5 8/26/2009 2:08:04 PM Good
58 DataSim:Sine -0.29389262621147 8/26/2009 2:08:06 PM Good
59 DataSim:Sine 1.82247325188E-13 8/26/2009 2:08:07 PM Good
60 DataSim:Sine -0.404508497179667 8/26/2009 2:08:08 PM Good
Record: 55 of 60

A different kind of table might have one column for a timestamp, and then additional columns containing the values of different VINE points logged at each time, like this:



A database table may require each row (or record) to be uniquely identified. This is commonly done through a *key column*, whose main and (sometimes only) purpose is to provide a unique identifying value to the row. Most DBMSs allow this value to be assigned manually by the database user, or automatically through an *auto-incrementing counter* or other mechanism. It is possible for a table to have multiple key columns, but some funtions in Skkynet VINE will only write to tables with a single key column. For more information about configuring database tables with Skkynet VINE, please refer to Section 7.5, *Configuring a Database Table*

Connecting to a Database: ODBC

Connecting to a database is done through the DBMS, which normally offers two possibilities: *native drivers* and *ODBC* (Open Database Connectivity). Native drivers are inconvenient to use because each requires its own programming interface. ODBC, on the other hand, specifies a standardized, common interface that is available from almost every database vendor, including SQL Server, MS Access, MySQL, Oracle, and many, many more. Skkynet VINE uses ODBC to connect to databases.

ODBC supports communication with a DBMS locally or across a network, using an *ODBC driver*. Every ODBC-compliant DBMS provides an ODBC driver, which needs to be installed on the user's machine. For example, there is an ODBC driver for MS Access, for SQL Server, for MySQL, and so on.

You can use the Windows ODBC Data Source Administrator to configure a connection between the ODBC driver and the specific database you want to work with. That configuration is called the *Data Source Name*, or *DSN*. For example, Skkynet VINE references the DSN and uses the configured connection for the ODBC driver to connect to the database.

Configuring the DSN is straightforward, varying slightly depending on the ODBC driver you are working with. Usually you need to select an ODBC driver, create a name for the DSN, and select a database. Other information, such as a login name or password may be required or optional. For more information, please refer to Section 8.3, *Setting up the DSN (Data Source Name)*

Accessing Data: SQL

Once connected to a database, any *queries* (requests) to retrieve, modify, add, or delete data must be made through a language. The most popular database query language is SQL (Structured Query Language), pronounced "sequel" or "ess-kyu-el". Created in the 1960s, this language has become a widely-used standard supported by most DBMSs, although there are some minor variations in certain commands offered.

Skkynet VINE uses SQL to write to and read from databases. When you configure the Data Logging interface to write VINE point values, under the hood the commands used are written in SQL. VINE scripts also use SQL commands to write and read data. For example, the following line from the `ODBCTutorial3.g` script uses an SQL **SELECT** command to pull all of the data from a database table.

```
result = .conn.QueryToClass (.tableclass, string ("select * from ", .tablename));
```

The **SELECT** command is often used with the `FROM` and `WHERE` operators, in queries such as this:

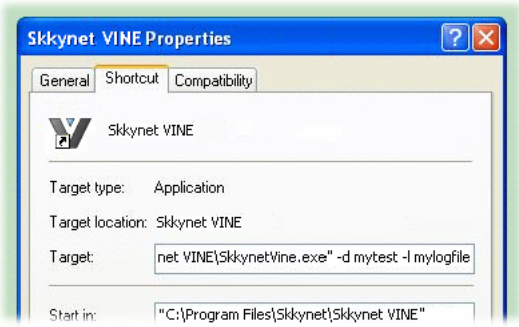
```
SELECT data_element_1
FROM table_1
WHERE data_element_1 > 32 and data_element_1 < 212;
```

The syntax of SQL is fairly simple, and there are many books and online tutorials that can help you learn. The information presented here about SQL, ODBC, and databases in general should be enough to get started logging data with Skkynet VINE.

Appendix D. Command Line Options

For more control of Skkynet VINE on startup, you can use its command line options. These options let you specify data domains, ports, and several other configuration items each time VINE starts. Except where noted, these options apply to Cascade DataHub in Windows, Linux, or QNX, as well as Cascade Connect. You need to enter these options in the Properties section of the VINE shortcut icon.

1. Right click on the VINE shortcut icon and select Properties. The Skkynet VINE Properties window will appear, with the Shortcut tab selected:



2. Enter the options in the Target field, after the quotation marks, as illustrated above. The available options are as follows:

`-a`

Transmit all point messages to all registered clients, even if the value does not change.

`-b` *`size`*

The maximum message buffer size.

`-d` *`domain`*

The domain name for this VINE. This option can be used multiple times to get multiple domains on a single VINE.

`-D`

Do not detach from the controlling tty. Normally VINE will detach itself and become immune to interrupts and termination on the controlling tty. If this option is used, then an *&* is necessary to run **datahub** in the background.

`-f` *`file`*

Load this configuration file.

`-h`

Print a help message showing a summary of all these arguments.

`-H` *`home_path`*

The full path to the directory that will contain the configuration and license files. This takes precedence over `-U`. If the directory cannot be found or created, the files will be stored in the installation directory.

`-I`

Hide the system tray icon when the DataHub starts. Only works in Windows.

`-l` *`file`*

Log messages to this file.

`-m` *`port`*

Acting as a TCP slave, attach to a TCP master on this port or service. The port is the matching port number of the master, usually `4502`;

`-M` *`address`*

Acting as a TCP slave, attach to a TCP master on this host. The address is a machine name, such as `developers.cogentrts.com` or a machine address, such as `192.168.3.15`.

`-n` *`domain`*

Acting as a TCP slave, tunnel/mirror this domain from the TCP master. The named domain on the master will be tunneled/mirrored to a domain of the same name in the slave.

`-p` *`port`*

Act as a TCP master and listen on this port/service.

`-P`

Show the properties window when the DataHub starts. Only works in Windows.

`-q` *`queue`*

Specify an alternate queue name for this VINE. Normally **datahub** chooses its own queue name to be unique on the network.

`-s`

Synchronized: VINE will ignore changes to a point if the point's current timestamp is more recent.

`-t`

Automatically generate a timestamp on unstamped points.

`-U`

VINE should NOT create a directory within the user's personal `Application Data` directory to store the configuration and license files, but rather in the application installation directory. This has a lower precedence than `-H`.

`-v`

Generate copious debugging information to the standard output. (Implies use of `-D`).

`-V`

Print the version number.

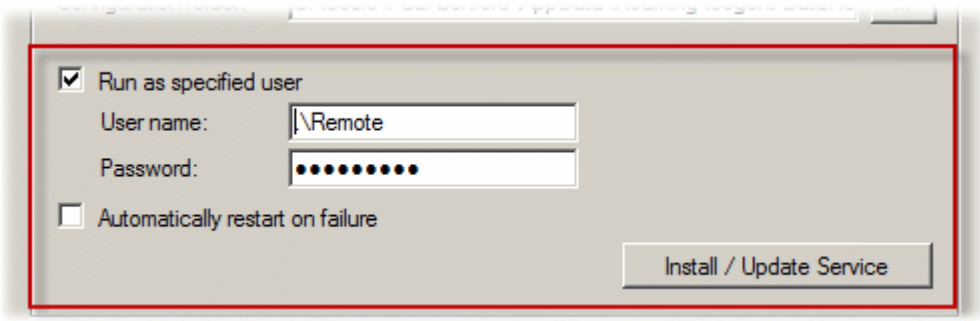
`-X`

Exit immediately (usually used with `-V`).

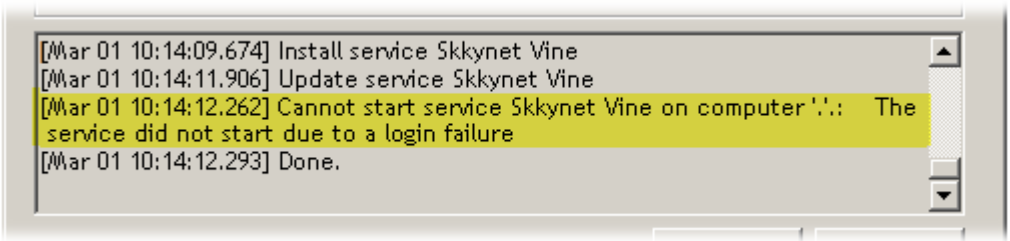
3. Click OK and restart Skkynet VINE. The options you have chosen should take effect. Keep in mind that if your configuration file has different values for these options, it will override what you have entered here.

Appendix E. Running as a Windows Service (Specified User)

The following information relates to running VINE service as a specified user. **We recommend not doing this as the properties are only available on the service console of the SYSTEM user account.** But in some cases this may be unavoidable.



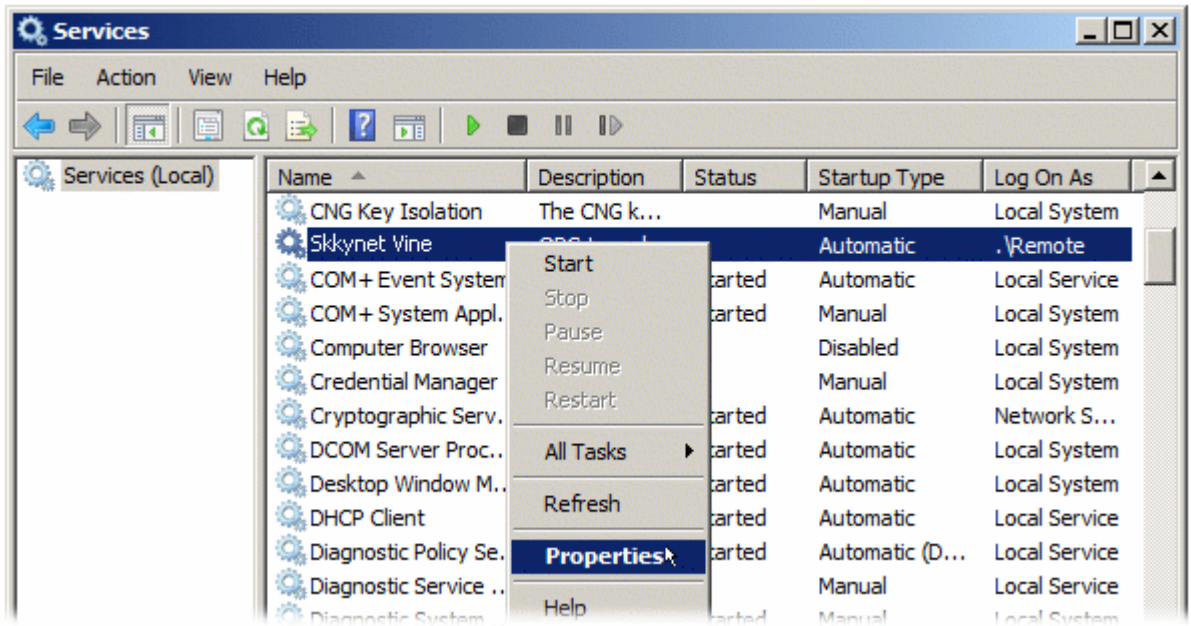
If you try to install the DataHub service as a specified user as explained in Section 1.3, *Installing VINE as a Service* then you may find that the service does not run, and you get an error message related to a logon failure, such as this:


[Mar 01 10:14:09.674] Install service Skkynet Vine
[Mar 01 10:14:11.906] Update service Skkynet Vine
[Mar 01 10:14:12.262] Cannot start service Skkynet Vine on computer '.'.: The service did not start due to a login failure
[Mar 01 10:14:12.293] Done.

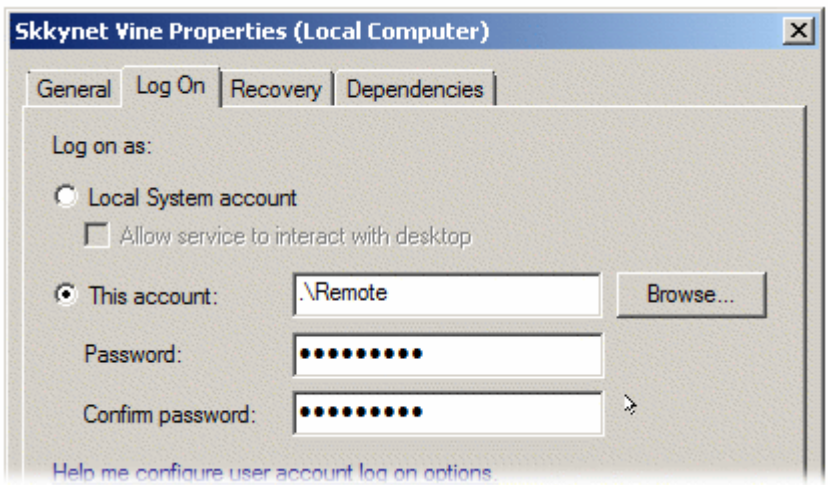
This error occurs when a user who is going to run the VINE service does not have the permission, or right, to log on as a service in Windows.

One way to set this permission/right is as follows:

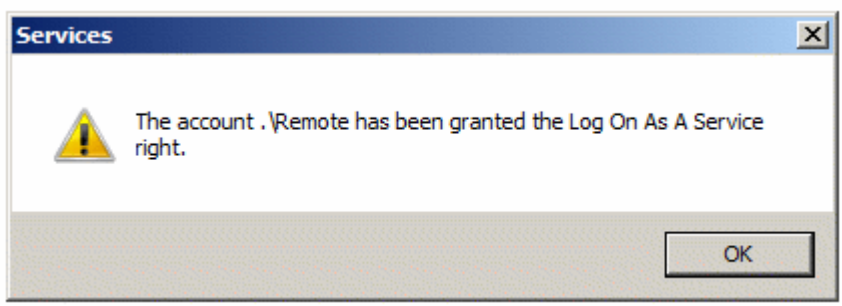
1. Open the Windows Services window from the Control Panel by selecting Administrative Tools and then Services.
2. Right click on Skkynet VINE and select Properties.



3. Click on the Log On tab and then retype the password for the user account you are trying to use. Then click Apply.


Skkynet Vine Properties (Local Computer)
General Log On Recovery Dependencies
Log on as:
Local System account
Allow service to interact with desktop
This account: .\Remote Browse...
Password: ••••••••••
Confirm password: ••••••••••
Help me configure user account log on options.

4. You should see a Windows message acknowledging that the right to log on as a service has been granted.



5. Now if you try to start the service from the Skkynet VINE Services Manager again it should work.

Appendix F. Windows Services File for Tunnel/Mirror

If you enter a name for the Master service/port in the Tunnel/Mirror tab of the Properties window, that name must be listed in the Windows `services` file.

Finding the `services` file

Since each manufacturer's `services` file is different, you must find the `services` file that your TCP/IP protocol stack is currently using. A Microsoft TCP/IP implementation typically puts the `services` file in the `C:\Windows` (or equivalent) directory. Most third party software either installs the `services` file in the same directory that their software was installed or into a directory named `C:\ETC`. Refer to your TCP/IP documentation for the location of this file.

- In Windows NT, the installation program attempts to edit a `services` file in the `\winnt\system32\drivers\etc` directory.
- In Windows XP, the installation program currently does not attempt to edit the `services` file. The default directory for that file is `C:\WINDOWS\system32\drivers\etc\`.

Editing the `services` file

Once you have found the `services` file, you must add the line:

```
service_name    ####/TCP
```

using a text editor. For example, to assign the name `vine` to port `4502`, you would add the line:

```
vine     4502/TCP
```

Remember that if you edit the services file with Notepad, it will attach a `.txt` suffix when it saves the file so that you will not in fact have edited the system services file, but instead created a new file, named `services.txt`. You should rename that file `services`, without the `.txt` extension.

Appendix G. Error Messages

This section presents error numbers that the developer may encounter when using Skkynet VINE.

G.1. Windows Error Numbers

Number	Error String	Error Description
0	EOK	No error.
1	EPERM	No permissions, or the user is not the process owner.
2	ENOENT	No such file or directory.
3	ESRCH	No such process.
4	EINTR	Interrupted system call.
5	EIO	I/O error.
6	ENXIO	No such device or address.
7	E2BIG	Argument list is too big.
8	ENOEXEC	Executable format is not recognized.
9	EBADF	Bad file number or invalid file descriptor.
10	ECHILD	No child processes exist.
11	EAGAIN	Resource temporarily unavailable or operation would block.
12	ENOMEM	Out of memory.
13	EACCES	Permission denied.
14	EFAULT	Bad memory address.
15	ENOTBLK	Block operation attempted on non-block device.
16	EBUSY	Device or resource busy, or operation already in progress.
17	EEXIST	File exists.
18	EXDEV	Cross-device link.
19	ENODEV	No such device.
20	ENOTDIR	Not a directory.
21	EISDIR	Is a directory.
22	EINVAL	Invalid argument.
23	ENFILE	File table overflow.
24	EMFILE	Too many open files.
25	ENOTTY	Character operation on non-character device.
26	ETXTBSY	Text file is busy.
27	EFBIG	File is too large.
28	ENOSPC	No space left on device.
29	ESPIPE	Illegal seek attempted on a pipe.
30	EROFS	Attempted write to a read-only file system.
31	EMLINK	Too many links.
32	EPIPE	Broken pipe.
33	EDOM	Math argument out of data domain of function.
34	ERANGE	Result too large.

Number	Error String	Error Description
35	EUCLEAN	
36	EDEADLOCK	Deadlock avoided.

G.2. Windows TCP Error Numbers

Number	Error String	Error Description
10035	EWOULDBLOCK	Resource temporarily unavailable or operation would block.
10036	EINPROGRESS	Operation now in progress.
10037	EALREADY	Device or resource busy, or operation already in progress.
10038	ENOTSOCK	Socket operation on non-socket.
10039	EDESTADDRREQ	Destination address required.
10040	EMSGSIZE	Message too long.
10041	EPROTOTYPE	Protocol wrong type for socket.
10042	ENOPROTOOPT	Protocol not available.
10043	EPROTONOSUPPORT	Protocol not supported.
10044	ESOCKTNOSUPPORT	Socket type not supported.
10045	EOPNOTSUPP	Operation not supported.
10046	EPFNOSUPPORT	Protocol family not supported.
10047	EAFNOSUPPORT	Address family not supported by protocol family.
10048	EADDRINUSE	Address already in use.
10049	EADDRNOTAVAIL	Can’t assign requested address.
10050	ENETDOWN	Network is down.
10051	ENETUNREACH	Network is unreachable.
10052	ENETRESET	Network dropped connection on reset.
10053	ECONNABORTED	Software caused connection abort.
10054	ECONNRESET	Connection reset by peer.
10054	ECONNRESET	Connection reset by peer.
10055	ENOBUFS	No buffer space available.
10056	EISCONN	Socket is already connected.
10057	ENOTCONN	Socket is not connected. Note: If this error appears with this message: TCP master service initialization failed: 10057 it could mean that a program is holding open port `4502`. This is may be caused by VINE not shutting down properly for some reason and it is still running, even though the icon is not showing. To check, look in the Windows Task List and see if you can see a VINE running. If it is, then you can kill it in the list and restart VINE. If it’s not in the list, then you need to use a firewall program to check to see which program is using port 4502.

Number	Error String	Error Description
10058	ESHUTDOWN	Can’t send after socket shutdown.
10059	ETOOMANYREFS	Too many references: can’t splice.
10060	ETIMEDOUT	Connection timed out.
10061	ECONNREFUSED	Connection refused.
10062	ELOOP	Too many symbolic link or prefix loops.
10063	ENAMETOOLONG	Name too long.
10064	EHOSTDOWN	Host is down.
10065	EHOSTUNREACH	No route to host.
10066	ENOTEMPTY	Directory not empty.
10067	EPROCLIM	Process count limit reached.
10068	EUSERS	User count limit reached.
10069	EDQUOT	Quota limit reached.
10070	ESTALE	Potentially recoverable i/o error.
10071	EREMOTE	Too many levels of remote in path.

G.3. Windows DDE Error Numbers

Number	Error String	Error Description
16384	DMLERR_ADVACKTIMEOUT	Timeout waiting for an advise acknowledge.
16385	DMLERR_BUSY	Recipient is busy.
16386	DMLERR_DATAACKTIMEOUT	Timeout waiting for an advise data acknowledge
16387	DMLERR_DLL_NOT_INITIALIZED	DDEML.DLL is not initialized.
16388	DMLERR_DLL_USAGE	General DDE library usage error.
16389	DMLERR_EXECACKTIMEOUT	Timeout waiting for an exec acknowledgment.
16390	DMLERR_INVALIDPARAMETER	Invalid parameter to DDEML function call.
16391	DMLERR_LOW_MEMORY	Memory is becoming low.
16392	DMLERR_MEMORY_ERROR	Memory is exhausted.
16393	DMLERR_NOTPROCESSED	Receiving task was not interested in message.
16394	DMLERR_NO_CONV_ESTABLISHED	No DDE conversation could be established.
16395	DMLERR_POKEACKTIMEOUT	Timeout waiting for a poke acknowledge.
16396	DMLERR_POSTMSG_FAILED	Attempt to post a window message failed.
16397	DMLERR_REENTRANCY	The DDE library was re-entered during a blocking call.
16398	DMLERR_SERVER_DIED	DDE server has died.
16399	DMLERR_SYS_ERROR	A DDE call has caused a system error.
16400	DMLERR_UNADVACKTIMEOUT	Timeout waiting for an unadvised acknowledge.

Number	Error String	Error Description
16401	DMLERR_UNFOUND_QUEUE_ID	DDE queue id could not be found.

Appendix H. Third-Party Source Licenses

The following licenses are being used in this product:

License for Scintilla and SciTE

Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

License for ScintillaNET

ScintillaNET is based on the Scintilla component by Neil Hodgson.

ScintillaNET is released on this same license.

The ScintillaNET bindings are Copyright 2002-2006 by Garrett Serack < <gserack@gmail.com>>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

GARRETT SERACK AND ALL EMPLOYERS PAST AND PRESENT DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL GARRETT SERACK AND ALL EMPLOYERS PAST AND PRESENT BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

SHTTPD

Copyright (c) 2004-2005 Sergey Lyubka <valenok@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Appendix I. Media Source Licenses

Some of the images, sound files, and videos distributed with this product are covered by one or more of the following licenses or terms of use:

- GNU Lesser General Public License (LGPL)
- Creative Commons Attribution 2.5 Generic license
- Creative Commons Attribution-Share Alike 3.0 Unported license
- Sound Jay Terms of Use

Appendix J. GNU General Public License

GNU General Public License

Version 2, June 1991

Copyright © 1989, 1991 by Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

* *Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.*

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program",

below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE

PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type “show w”. This is free software, and you are welcome to redistribute it under certain conditions; type “show c” for details.

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than “show w” and “show c”; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program “Gnomovision” (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix K. GNU Lesser General Public License

GNU Lesser General Public License

Version 2.1, February 1999

Copyright © 1991, 1999 by Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

* *Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.*

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method:

1. we copyright the library, and
2. we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author’s reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the *Lesser* General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

Section 1

You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.

b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

   (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

Section 4

You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

Section 5

A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

Section 6

As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work

during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

Section 7

You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

Section 8

You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who

have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 9

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

Section 10

Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

Section 11

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 12

If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 13

The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

Section 14

If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 15

BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 16

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the library’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library ’Frob’ (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990 Ty Coon, President of Vice

That’s all there is to it!

I. Skkynet VINE Command Set

Table of Contents

acksuccess 238
add 239
alias 240
alive 241
append 242
assembly 243
attribute 244
auth 245
authreload 246
auto_create_domains 247
auto_timestamp 248
bridge 249
bridge_remove 251
bridge_remove_pattern 252
bridge_transform 253
cforce 255
cread 256
create 257
create_domain 258
creport 259
cset 260
cwrite 261
DDEAdvise 263
DDEConnect 264
DDEDisconnect 265
DDEExcelUnicode 266
DDEInit 267
DDEService 268
DDEUnadvise 269
DDEUnadvisePattern 270
DDEUnadvisePoint 271
debug 272
defaultprop 273
delete 274
div 275
domain 276
domains 277
domain_bridge 278
domain_bridge_enable 280

domain_bridge_prefer 281
domain_bridge_refresh 282
domain_bridge_remove 283
dump 284
enable_bridging 285
enable_dde_client 286
enable_dde_server 287
enable_domain_bridging 288
enable_scripting 289
error 290
execute_plugin 291
exit 292
force 293
format 294
get_client_stats 295
heartbeat 296
HistorianAdd 297
HistorianFlags 298
HistorianRemove 299
HistorianSaveConfig 300
HistorianSetConfiguring 301
HistoryGroupAdd 302
HistoryGroupAddPoint 303
HistoryGroupDeadband 304
HistoryGroupDefault 305
HistoryGroupFileTimes 306
HistoryGroupFlushTimes 307
HistoryGroupRemove 308
HistoryGroupStorageTimes 309
ignore 310
ignore_old_data 311
include 312
instance 313
load_config_files 314
load_plugin 315
load_scripts 316
lock 317
log_file 318
log_file_max 319
log_to_file 320
mirror_master 321
mirror_master_2 322
mult 324

private_attribute **325**
property **326**
quality **327**
read **328**
report **329**
report_domain **330**
report_errors **331**
request_initial_data **332**
save_config **333**
secure **334**
set **335**
set_access **336**
set_authoritative **337**
set_canonical **338**
show_data **339**
show_debug_messages **340**
show_event_log **341**
show_icon **342**
show_properties **343**
show_script_log **344**
subassembly **345**
success **346**
tcp_service **347**
timeout **348**
TunnelEnable **349**
TunnelEnablePlain **350**
TunnelEnableSlave **351**
TunnelEnableSSL **352**
TunnelPlainPort **353**
TunnelSlaveStatus **354**
TunnelSSLCert **355**
TunnelSSLPort **356**
transmit_insignificant **357**
type **358**
unload_plugin **359**
unreport **360**
unreport_domain **361**
version **362**
write **363**

This is the internal command set for VINE. For general information on how to use these commands, please refer to Chapter 17, *Using VINE Commands*.

acksuccess

`acksuccess` — tells the VINE to return success messages.

Syntax

```
(acksuccess 0|1)
```

Arguments

`0|1`

Use `1` to have messages returned, or `0` to not have messages returned.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command tells the VINE to return a message for successfully executed command. Error messages are always sent for unsuccessful attempts to execute a command. Please refer to Return Syntax for details about the message format.

add

add — adds a value to a point.

Syntax

```
(add name number [secs] [nano])
```

Arguments

name

The name of a point, which must be a number type.

number

A value to add to the value of the point.

secs

The time in seconds. If this is not specified, the current date and time in seconds is used.

nano

A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to add to the value of a point.

alias

`alias` — creates an alias point for an existing point.

Syntax

`(alias` *`point alias`*`)`

Arguments

`point`

The name of a VINE point, as a string.

`alias`

A name for the new point, as a string.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a new point that will alias, or tunnel/mirror its value with, an existing point. The first point must exist, and the second point will be created if it does not exist. They do not need to be in separate domains.

alive

`alive` — tells the Cascade DataHub that the client is running.

Syntax

```
(alive)
```

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command provides a means for the client to tell the Cascade DataHub that it is still up and running.

append

append — appends a string to the value of a point.

Syntax

```
(append name string [secs] [nano])
```

Arguments

name

The name of a point, which must be a string type.

string

A string to add to the current string value of the point.

secs

The time in seconds. If this is not specified, the current date and time in seconds is used.

nano

A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to add a string to the value of a point.

assembly

`assembly` — creates an assembly.

Syntax

```
(assembly domain name [supername])
```

Arguments

`domain`

The name of the domain in which this assembly will be created.

`name`

A name for this assembly.

`supername`

The name of an assembly.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates an assembly level of a data organization. The *`supername`* denotes a special assembly that can be created to hold properties and attributes that may be used by several assemblies, in much the way that a parent class has instance variables that are used by child classes. For more information about assemblies and an example, please refer to Section 13.4.2, *Assemblies, Subassemblies, Attributes, and Properties*.

attribute

`attribute` — creates an attribute.

Syntax

```
(attribute domain assemblyname attrname typename)
```

Arguments

`domain`

The domain in which this attribute applies.

`assemblyname`

The assembly or subassembly in which this attribute applies.

`attrname`

The name of this attribute.

`typename`

The type of this attribute.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates an attribute. For more information and an example, please refer to Section 13.4.2, *Assemblies, Subassemblies, Attributes, and Properties*.

auth

`auth` — requests authentication for a client.

Syntax

```
(auth username password)
```

Arguments

`username`

A user name in plain text. Any non-alphanumeric characters must be in double quotes.

`password`

A password in plain text. Any non-alphanumeric characters must be in double quotes.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is sent to the VINE by a client to request authentication.

authreload

`authreload` — is new, not yet documented.

Syntax

```
(authreload)
```

Description

This command has not yet been documented.

auto_create_domains

`auto_create_domains` — automatically adds domains requested by clients.

Syntax

```
(auto_create_domains 0|1)
```

Arguments

`0|1`

Use `1` to have domains added, or `0` to not have domains added automatically.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command instructs the VINE to create a domain automatically if a client requests a domain that doesn't already exist. This corresponds to the Automatically add domains requested by clients checkbox in the General option of the Properties window.

auto_timestamp

`auto_timestamp` — adds timestamps to unstamped changes.

Syntax

```
(auto_timestamp 0|1)
```

Arguments

`0|1`

Use `1` to add timestamps, or `0` to not add timestamps automatically.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command instructs the VINE to timestamp points automatically to changes that don’t already have a timestamp. This corresponds to the Automatically add a timestamp to unstamped changes checkbox in the General option of the Properties window.

bridge

`bridge` — creates a bridge between two points.

Syntax

```
(bridge source destination [flags [multiply add [srcmin srcmax dstmin dstmax]]])
```

Arguments

`source`

The fully qualified point name of the source, or starting point of the bridge.

`destination`

The fully qualified point name of the destination, or ending point of the bridge.

`flags`

A bitwise combination of:

`1`	Forward bridge: bridge from source to destination
`2`	Inverse bridge: bridge from destination to source
`16`	Clamp output to the minimum. (Range mapping only.)
`32`	Clamp output to the maximum. (Range mapping only.)
`256`	The bridge is a direct copy
`512`	The bridge uses a linear transformation
`1024`	The bridge uses range mapping
`4096`	The bridge is disabled

Bits `256`, `512` and `1024` are mutually exclusive.

`multiply`

The multiplier value for a linear transformation. This is ignored if `(flags & 512) == 0`.

`add`

The adder value for a linear transformation. This is ignored if `(flags & 512) == 0`.

`srcmin`

The minimum range map value for the source point. This is ignored if `(flags & 1024) == 0`.

`srcmax`

The maximum range map value for the source point. This is ignored if `(flags & 1024) == 0`.

`dstmin`

The minimum range map value for the destination point. This is ingored if `(flags & 1024) == 0`.

`dstmax`

The maximum range map value for the destination point. This is ingored if `(flags & 1024) == 0`.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a bridge between two data points so that a change to the value of one point automatically propagates to the other point. The scaling and the limits on source and destination points used for linear transformations are stored with the bridge so that if you decide to change from a direct bridge to one that uses linear transformations your previous entries are preserved. The values themselves are only applied when the flag set indicates the corresponding transfer function.

bridge_remove

`bridge_remove` — deletes a bridge.

Syntax

```
(bridge_remove source destination)
```

Arguments

`source`

A string containing the fully qualified point name of the source, or starting point of the bridge.

`destination`

A string containing the fully qualified point name of the destination, or ending point of the bridge.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command permanently deletes a bridge. The *`source`* and *`destination`* names must be fully qualified, specifying the domain and point name, as in "*`domain`*:*`pointname`*".

bridge_remove_pattern

`bridge_remove_pattern` — deletes all bridges that match a pattern.

Syntax

```
(bridge_remove_pattern source_pattern destination_pattern)
```

Arguments

`source_pattern`

A string containing a pattern for the name of the source points of the bridge, such as `"domain1:*"`.

`destination_pattern`

A string containing a pattern for the name of the destination points of the bridge, such as `"*:*"`.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command permanently deletes all bridges that match a specified pattern. The available patterns are as follows:

- ***** matches any number of characters, including zero.
- **[*c*]** matches a single character which is a member of the set contained within the square brackets.
- **[^*c*]** matches any single character which is not a member of the set contained within the square brackets.
- **?** matches a single character.
- **{*xx*,*yy*}** matches either of the simple strings contained within the braces.
- ***c*** (a backslash followed by a character) - matches that character.

bridge_transform

`bridge_transform` — modifies an existing bridge.

Syntax

```
(bridge_transform name flags [multiply add [srcmin srcmax dstmin dstmax]])
```

Arguments

`name`

The name of the bridge.

`flags`

A bitwise combination of:

`1`	Forward bridge: bridge from source to destination
`2`	Inverse bridge: bridge from destination to source
`16`	Clamp output to the minimum. (Range mapping only.)
`32`	Clamp output to the maximum. (Range mapping only.)
`256`	The bridge is a direct copy
`512`	The bridge uses a linear transformation
`1024`	The bridge uses range mapping
`4096`	The bridge is disabled

Bits `256`, `512` and `1024` are mutually exclusive.

`multiply`

The multiplier value for a linear transformation. This is ignored if `(flags & 512) == 0`.

`add`

The adder value for a linear transformation. This is ignored if `(flags & 512) == 0`.

`srcmin`

The minimum range map value for the source point. This is ignored if `(flags & 1024) == 0`.

`srcmax`

The maximum range map value for the source point. This is ignored if `(flags & 1024) == 0`.

`dstmin`

The minimum range map value for the destination point. This is ingored if `(flags & 1024) == 0`.

`dstmax`

The maximum range map value for the destination point. This is ingored if `(flags & 1024) == 0`.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command modifies an existing bridge between two data points. The scaling and the limits on source and destination points used for linear transformations are stored with the bridge so that if you decide to change from a direct bridge to one that uses linear transformations your previous entries are preserved. The values themselves are only applied when the flag set indicates the corresponding transfer function.

cforce

`cforce` — creates a point and forces a value to be written to it.

Syntax

```
(cforce name value [confidence])
```

Arguments

`name`

The name of the point. This is a string.

`value`

A string representation of the value for the point. It will be interpreted into the type specified by the type parameter.

`value`

A string representation of the value for the point. The point value will be interpreted as integer, float or string based on the contents of the value string. This function tries each type in order, and uses the first type for which the value parameter is a valid representation. Double quotes around the value parameter are ignored. For example:

- `123` is an integer.
- `123.4` is a float.
- `1.234e2` is a float.
- `"123"` is an integer.
- `123abc` is a string.

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (`\`) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with `\n`, `\r`, `\f`, `\t` respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is the same as **cset**, except that it forces a write even if the VINE would otherwise refuse it, for example if the point is old, the value is insignificant or hasn't changed, or the point is marked as read-only. When this value is set, the following attributes of the point are set as follows:

- `seconds` and `nanoseconds` are set to the current time on the machine running the DataHub.
- `locked`, `sec`, and `quality` are all maintained at their previous values for this point.
- `flags` is set to `0`.

Please refer to the **write** command for more information about these parameters. See also **set** and **force**.

cread

`cread` — creates and reads a point.

Syntax

```
(cread name)
```

Arguments

`name`

The name of the point.

Returns

The complete point definition in a message, with this syntax:

```
(point name type value [conf security locked seconds nanoseconds flags quality])
```

where:

`name`

The name of the point.

`type`

The data type of the point, one of integer, floating point, or character string.

`value`

The value of the point.

`conf`

The confidence level of the point, `0` - `100` percent, unused by most applications.

`security`

The security level of the point, `0` to `32768`, where higher numbers represent higher security.

`locked`

`0` for locked, or `1` for unlocked.

`seconds`

The operating system time in seconds when the point was read.

`nanoseconds`

The number of nanoseconds after *`seconds`* when the point was read.

`flags`

User-defined flags.

`quality`

A constant representing a quality of the connection, assigned by the VINE for this point, such as `Good`, `Bad`, `Last known`, `Local override`, etc. The possible values are those supported by OPC in Microsoft Windows.

Description

This command creates a point and then reads the information it contains. See also **read**.

create

`create` — creates a new point.

Syntax

```
(create name [0|1])
```

Arguments

`name`

The name of the point, as a string.

`0|1`

Tells **create** what to do if a point already exists with that name. Use `1` to ignore an existing point and do nothing. Use `0` to have **create** throw an error. If nothing is entered, the default is `0`.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a new point in VINE. Normally it is not necessary to create points manually—VINE creates a new point any time a program sends one. However, this command is useful for creating points programmatically from within VINE. See also cset.

Example

Using the Gamma datahub_command function, you could pass a `create` command to VINE as follows to create `MyNewPoint` in the `default` domain, and assign it a value of `1`.

```
datahub_command ("(create default:MyNewPoint)", 1);
```

create_domain

`create_domain` — creates a new domain.

Syntax

```
(create_domain name)
```

Arguments

`name`

The name of the domain, as a string.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a new domain in the VINE. This corresponds to using the Add button in the Domains section in the General option of the Properties window.

creport

creport — creates a point and requests notification of changes.

Syntax

```
(creport name)
```

Arguments

The name of the data point to be created.

Returns

A message with the complete definition of the point.

Description

This command creates a data point and then requests the VINE to report changes (also called exceptions) to the value or any other information about the point, as soon as any change takes place. See also **report**.

cset

`cset` — creates a point and assigns it a value.

Syntax

```
(cset name value [confidence])
```

Arguments

`name`

The name of the point. This is a string.

`value`

A string representation of the value for the point. The point value will be interpreted as integer, float or string based on the contents of the value string. This function tries each type in order, and uses the first type for which the value parameter is a valid representation. Double quotes around the value parameter are ignored. For example:

- `123` is an integer.
- `123.4` is a float.
- `1.234e2` is a float.
- `"123"` is an integer.
- `123abc` is a string.

`confidence`

A confidence factor in the range of `0` to `100` (optional). This is not used by the VINE, so is available to programs that produce graduated confidence, such as expert systems. If this value is not specified, it is set to `100`.

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (`\`) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with `\n`, `\r`, `\f`, `\t` respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This is a convenience command that combines **create** and **set**, allowing you to create a point and set its value in a single command. When this value is set, the following attributes of the point are set as follows:

- `seconds` and `nanoseconds` are set to the current time on the machine running the DataHub.
- `locked`, `sec`, and `quality` are all maintained at their previous values for this point.
- `flags` is set to `0`.

Please refer to the **write** command for more information about these parameters. See also **set**, **force**, and **cforce**.

cwrite

`cwrite` — creates a point and writes information to it.

Syntax

```
(cwrite name type value conf sec locked seconds nanoseconds [flags quality])
```

Arguments

`name`

The name of the point. This is a string.

`type`

The type of point. One of

- `0` = string
- `1` = float (8-byte double)
- `2` = integer (32-bit integer)

`value`

A string representation of the value for the point. It will be interpreted into the type specified by the *`type`* parameter.

`conf`

A confidence factor in the range of `0` to `100`. This is not used by the VINE, so is available to programs that produce graduated confidence, such as expert systems.

`sec`

A security level for this point. This is rarely used. If a point's security level is set to a non-zero value then attempts to write to that point must claim a security level equal to or greater than the security level of the point. This uses a "good citizen" model - the writer can claim any security it wants, and is assumed to be honest - so there is no strong security here. It is intended for systems that want to avoid accidental changes to values. Security level can be from `0` to `32767`.

`locked`

An indication that the point is locked, and cannot be changed. Can be `0` or `1`. Attempts to write to a locked point will fail.

`seconds`

The UNIX epoch - seconds since Jan. 1, 1970, as produced by the `time()` function.

`nanoseconds`

The number of nanoseconds inside this second. Cannot exceed `1,000,000,000`.

`flags`

User level code should always send a `0` for this value.

`quality`

A quality indicator consistent with the OPC DA specification. This is not a bit field. It can be one of:

`PT_QUALITY_BAD`	`0`
`PT_QUALITY_UNCERTAIN`	`0x40`
`PT_QUALITY_GOOD`	`0xc0`

```
PT_QUALITY_CONFIG_ERROR             0x04
PT_QUALITY_NOT_CONNECTED            0x08
PT_QUALITY_DEVICE_FAILURE           0x0c
PT_QUALITY_SENSOR_FAILURE           0x10
PT_QUALITY_LAST_KNOWN               0x14
PT_QUALITY_COMM_FAILURE             0x18
PT_QUALITY_OUT_OF_SERVICE           0x1c
PT_QUALITY_WAITING_FOR_INITIAL_DATA 0x20

PT_QUALITY_LAST_USABLE              0x44
PT_QUALITY_SENSOR_CAL               0x50
PT_QUALITY_EGU_EXCEEDED             0x54
PT_QUALITY_SUB_NORMAL               0x58
PT_QUALITY_LOCAL_OVERRIDE           0xd8
```

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (\) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with `\n`, `\r`, `\f`, `\t` respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you manually write information to a point. If the point does not exist, the point is automatically created and the value is written. This command is identical to the **write** command, except that **write** will produce an error if the point does not exist.

DDEAdvise

`DDEAdvise` — sets up the item for a DDEAdvise connection.

Syntax

```
(DDEAdvise label item pointname [domain])
```

Arguments

`label`

A string that identifies the connection for this item.

`item`

The item name, as a string

`pointname`

The name of the point with which the *`item`* is associated, as a string.

`domain`

The domain of the point. If unspecified, it defaults to the `default` domain.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command supports a DDEAdvise connection. It associates a DDE item with a point and domain, as well as the label used by the item's service and topic (see **DDEConnect**). This command is used when you add or edit connections in the DDE option of the Properties window. For more information on DDE connections, please see Appendix A, *DDE Overview*.

DDEConnect

`DDEConnect` — makes a connection to a DDE service and topic.

Syntax

```
(DDEConnect label service topic [retry_sec])
```

Arguments

`label`

A string that identifies this connection.

`service`

The DDE service name, as a string

`topic`

The DDE topic, as a string.

`retry_sec`

The time interval, in seconds, that the connection should be retried in case of a disconnect or network failure.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command establishes a connection to a DDE service and topic, and names the connection. The point or points that use this connection are defined by the **DDEAdvise** command. Both of these commands are used when you add or edit connections in the DDE option of the Properties window. For more information on DDE connections, please see Appendix A, *DDE Overview*.

DDEDisconnect

`DDEDisconnect` — disconnects and discards a DDE connection.

Syntax

```
(DDEDisconnect label)
```

Arguments

`label`

A label identifying a connection, as assigned by the **DDEConnect** command.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command disconnects and discards a DDE connection that had been previously established by the **DDEConnect** command.

DDEExcelUnicode

`DDEExcelUnicode` — accepts Unicode characters in strings in Excel.

Syntax

```
(DDEExcelUnicode 0|1)
```

Arguments

`0|1`

`1` enables the capability, `0` disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This commandenables or disables the capability of the Cascade DataHub to function properly with Unicode characters in strings in Excel. This corresponds to the Accept non-English characters in Excel strings (slower) checkbox in the DDE option of the Properties window.>

DDEInit

`DDEInit` — initializes the VINE to act as a DDE server.

Syntax

```
(DDEInit)
```

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command instructs the VINE to act as a DDE server. Service names are assigned using the **DDEService** command.

DDEService

`DDEService` — assigns a DDE service name.

Syntax

```
(DDEService service)
```

Arguments

`service`

A DDE service name for the VINE, as a string.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command assigns a DDE service name to the Cascade DataHub. It is possible to use several names simultaneously. This command is used when adding service names to the VINE in the DDE option of the Properties window. Also see **DDEInit**.

DDEUnadvise

`DDEUnadvise` — removes an item from a DDE connection.

Syntax

```
(DDEUnadvise label item)
```

Arguments

`label`

A label identifying a connection, as assigned by the **DDEConnect** command.

`item`

The DDE item name of the point you wish to stop advising on.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command essentially undoes the **DDEAdvise** command, removing an item from a DDE connection.

DDEUnadvisePattern

`DDEUnadvisePattern` — removes multiple items from a DDE connection.

Syntax

`(DDEUnadvisePattern` *`label pattern`*`)`

Arguments

`label`

A label identifying a connection, as assigned by the **DDEConnect** command.

`pattern`

A pattern that matches the DDE item names of the points you wish to stop advising on.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command undoes the **DDEAdvise** command for a group of points that match the *`pattern`*, removing those items from a DDE connection. The available patterns are as follows:

- ***** matches any number of characters, including zero.
- **[*c*]** matches a single character which is a member of the set contained within the square brackets.
- **[^*c*]** matches any single character which is not a member of the set contained within the square brackets.
- **?** matches a single character.
- **{*xx*,*yy*}** matches either of the simple strings contained within the braces.
- ***c*** (a backslash followed by a character) - matches that character.

DDEUnadvisePoint

`DDEUnadvisePoint` — removes an item from a DDE connection, by its point name.

Syntax

```
(DDEUnadvisePoint label pointname)
```

Arguments

`label`

A label identifying a connection, as assigned by the**DDEConnect**.

`point`

The point name corresponding to the DDE item you wish to stop advising on.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command essentially undoes the **DDEAdvise** command, removing an item from a DDE connection.

debug

debug — sets the debug level.

Syntax

```
(debug debug_level)
```

Arguments

debug_level

An integer from `0` to `4` specifying the debug level.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you specify the level of detail of debugging, from the least (`0`) to the most (`4`).

defaultprop

`defaultprop` — sets a default type for a property.

Syntax

```
(defaultprop domain type propname)
```

Arguments

`domain`

The domain name of the property whose default type will be set.

`type`

The default type for this property, as a string.

`propname`

The name of the property, as a string.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets a default type for a property on a given domain. For more information and an example, please refer to Section 13.4.2, *Assemblies, Subassemblies, Attributes, and Properties*.

delete

`delete` — deletes a point—use with caution.

Syntax

```
(delete pointname [domain])
```

Arguments

`pointname`

The name of the point to delete, as a string.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

Deleting points with this command could cause unexpected behavior for other users of the point.

This command allows you to delete points from the VINE. You should exercise caution when using it, however, because any program connected to the VINE now or in the future might be relying on the existence of that point. We suggest you use this command only after ensuring that no connecting program uses that point.

div

`div` — does division on the value of a point.

Syntax

```
(append name number [secs] [nano])
```

Arguments

`name`

The name of a point, which must be a number type.

`number`

A value to divide the value of the point by.

`secs`

The time in seconds. If this is not specified, the current date and time in seconds is used.

`nano`

A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to divide to the value of a point.

domain

`domain` — identifies the client domain name.

Syntax

```
(domain domain_name)
```

Arguments

`domain_name`

The name of the domain.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command identifies to the Cascade DataHub the domain that the client is using.

domains

`domains` — lists all domains in the VINE.

Syntax

```
(domains)
```

Arguments

none

Returns

On success, a message with the following syntax:

```
(domains "default" "domain1" "domain2" "domain3" ...)
```

the following syntax. On error, an error message as described inReturn Syntax.

Description

This command generates a response message listing all of the domains currently in the VINE. The response message is different from the usual VINE command return syntax, in that it doesn't contain the string `success` on success.

domain_bridge

`domain_bridge` — configures a redundancy set.

Syntax

```
(domain_bridge label src1 src2 dst flags switchflags point quality value
 statepoint controlpoint statuspoint1 statuspoint2 [fill_delay_ms])
```

Arguments

Many of these parameters correspond to the similarly named entry field or checkbox in the Configure Redundancy dialog optons in the Redundancy option of the Properties window, shown here in parentheses.

`label`

A label for this redundancy set, as a string. (Label:)

`src1`

The domain for the first data source, as a string. (Source Domain 1:)

`src2`

The domain for the second data source, as a string. (Source Domain 2:)

`dst`

The name of the domain that will contain the output. (Output Domain)

`flags`

are not yet documented.

`switchflags`

are not yet documented.

`point`

The name of a point which will be used in comparisons. (For this point)

`quality`

The quality of the data which will be used in comparisons, as a number. The valid numbers and corresponding qualities can be found in **quality**. (Data quality is:)

`value`

The value of the data which will be used in comparisons, as a number. (Data value is:)

`statepoint`

The name of a point that will be used to indicate which data source is currently being used. (Point for current source number:)

`controlpoint`

The name of a point that will be used to indicate which data source is preferred. (Point for preferred source number:)

`statuspoint1`

The name of a point that will be used to indicate the current status of the first data domain. (Point for the current state of domain 1:)

`statuspoint2`

The name of a point that will be used to indicate the current status of the second data domain. (Point for the current state of domain 2:)

`fill_delay_ms`

The number of seconds to wait before switching domains if the data flow detection is activated.(Switch sources if data stops for:)

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command configures a VINE redundancy set. It corresponds to the Configure Redundancy dialog optons in the Redundancy option of the Properties window.

domain_bridge_enable

`domain_bridge_enable` — .

Syntax

```
(domain_bridge_enable label [enabled])
```

Arguments

`0|1`

`1` enables the capability, `0` disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This commandenables or disables the VINE Redundancy feature. It corresponds to the Redundancy checkbox in the Redundancy option of the Properties window.

domain_bridge_prefer

`domain_bridge_prefer` — is new, not yet documented.

Syntax

```
(domain_bridge_prefer label [index])
```

Description

This command has not yet been documented.

domain_bridge_refresh

`domain_bridge_refresh` — is new, not yet documented.

Syntax

```
(domain_bridge_refresh label)
```

Description

This command has not yet been documented.

domain_bridge_remove

`domain_bridge_remove` — is new, not yet documented.

Syntax

```
(domain_bridge_remove label)
```

Description

This command has not yet been documented.

dump

`dump` — writes the entire content of the VINE to a file.

Syntax

```
(dump filename)
```

Arguments

`filename`

The name of the file to write to.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command dumps the contents of the VINE to a file. The results are listed by domain, data types, and assemblies.

enable_bridging

`enable_bridging` — enables or disables bridging capabilities.

Syntax

```
(enable_bridging 0|1)
```

Arguments

`0|1`

`1` enables the capability, `0` disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the Cascade DataHub to support bridging.

See Also

bridge

enable_dde_client

`enable_dde_client` — enables or disables DDE client capabilities.

Syntax

```
(enable_dde_client 0|1)
```

Arguments

`0|1`

`1` enables the capability, `0` disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the Cascade DataHub to act as a DDE client. This corresponds to the Act as a DDE client... checkbox in the DDE option of the Properties window.

enable_dde_server

`enable_dde_server` — enables or disables DDE server capabilities.

Syntax

```
(enable_dde_server 0|1)
```

Arguments

`0|1`

`1` enables the capability, `0` disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the Cascade DataHub to act as a DDE server. This corresponds to the Act as a DDE server... checkbox in the DDE option of the Properties window.

enable_domain_bridging

`enable_domain_bridging` — enables or disables domain bridging capabilities.

Syntax

```
(enable_domain_bridging 0|1)
```

Arguments

`0|1`

`1` enables the capability, `0` disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command has not yet been documented.

enable_scripting

`enable_scripting` — enables or disables scripting capabilities

Syntax

```
(enable_scripting 0|1)
```

Arguments

`0|1`

`1` enables the capability, `0` disables it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command enables or disables the capability of the Cascade DataHub to support scripting.

error

`error` — sends an error with an error string.

Syntax

`(error` *`errstring`*`)`

Arguments

`errstring`

a string containing an error message.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sends an error to the VINE, which it displays in the Event Log, along with the message from the *`errstring`*.

execute_plugin

`execute_plugin` — executes a plugin. (*experimental*)

Syntax

```
(execute_plugin plugin_name)
```

Arguments

`plugin_name`

The name of a plugin.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the VINE to execute a plugin.

The commands related to plugins are currently experimental.

exit

`exit` — shuts down the VINE.

Syntax

```
(exit [exit_status])
```

Arguments

`exit_status`

An optional string to describe the circumstances of the shut-down.

Returns

A message containing the *`exit_status`* message.

Description

This command closes the VINE and all associated windows.

force

`force` — forces a write to a point.

Syntax

```
(force name value [confidence])
```

Arguments

`name`

The name of the point. This is a string.

`value`

A string representation of the value for the point. The point value will be interpreted as integer, float or string based on the contents of the value string. This function tries each type in order, and uses the first type for which the value parameter is a valid representation. Double quotes around the value parameter are ignored. For example:

- `123` is an integer.
- `123.4` is a float.
- `1.234e2` is a float.
- `"123"` is an integer.
- `123abc` is a string.

`confidence`

A confidence factor in the range of `0` to `100` (optional). This is not used by the VINE, so is available to programs that produce graduated confidence, such as expert systems. If this value is not specified, it is set to `100`.

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (`\`) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with `\n`, `\r`, `\f`, `\t` respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is the same as **set**, except that it forces a write even if the VINE would otherwise refuse it, for example if the point is old, the value is insignificant or hasn't changed, or the point is marked as read-only. When this value is set, the following attributes of the point are set as follows:

- `seconds` and `nanoseconds` are set to the current time on the machine running the DataHub.
- `locked`, `sec`, and `quality` are all maintained at their previous values for this point.
- `flags` is set to `0`.

Please refer to the **write** command for more information about these parameters. See also **set** and **cforce**.

format

`format` — is an efficiency enhancement for Linux.

Syntax

```
(format flag)
```

Arguments

`flag`

One of the following flags:

- `1` turns on the behavior.
- `0` turns off the behavior.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is an efficiency enhancement for Linux, for SRR Module connections only. It tells the VINE that this client would like to get point exceptions in binary instead of ASCII encoding. It is more CPU efficient but does not work well in a heterogeneous network.

get_client_stats

`get_client_stats` — is new, not yet documented.

Syntax

```
(get_client_stats [label_pattern])
```

Description

This command has not yet been documented.

heartbeat

`heartbeat` — establishes a heartbeat message.

Syntax

```
(heartbeat ms)
```

Arguments

`ms`

The number of milliseconds between each pulse.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command establishes a heartbeat message that verifies the connection every *`ms`* milliseconds. The heartbeat is sent from the VINE to the client.

HistorianAdd

`HistorianAdd` — is new, not yet documented.

Syntax

```
(HistorianAdd point [grouplabel])
```

Description

This command has not yet been documented.

HistorianFlags

`HistorianFlags` — is new, not yet documented.

Syntax

```
(HistorianFlags flags)
```

Description

This command has not yet been documented.

HistorianRemove

`HistorianRemove` — is new, not yet documented.

Syntax

```
(HistorianRemove point_pattern [group_pattern])
```

Description

This command has not yet been documented.

HistorianSaveConfig

`HistorianSaveConfig` — is new, not yet documented.

Syntax

```
(HistorianSaveConfig)
```

Description

This command has not yet been documented.

HistorianSetConfiguring

`HistorianSetConfiguring` — is new, not yet documented.

Syntax

```
(HistorianSetConfiguring 0|1)
```

Description

This command has not yet been documented.

HistoryGroupAdd

`HistoryGroupAdd` — is new, not yet documented.

Syntax

```
(HistoryGroupAdd group_label basedir extension cachesize flags)
```

Description

This command has not yet been documented.

HistoryGroupAddPoint

`HistoryGroupAddPoint` — is new, not yet documented.

Syntax

```
(HistoryGroupAddPoint group_label pointname)
```

Description

This command has not yet been documented.

HistoryGroupDeadband

`HistoryGroupDeadband` — is new, not yet documented.

Syntax

```
(HistoryGroupDeadband group_pattern flags absolute percent maxsecs maxcount)
```

Description

This command has not yet been documented.

HistoryGroupDefault

`HistoryGroupDefault` — is new, not yet documented.

Syntax

```
(HistoryGroupDefault label)
```

Description

This command has not yet been documented.

HistoryGroupFileTimes

`HistoryGroupFileTimes` — is new, not yet documented.

Syntax

```
(HistoryGroupFileTimes group_pattern days hours minutes)
```

Description

This command has not yet been documented.

HistoryGroupFlushTimes

`HistoryGroupFlushTimes` — is new, not yet documented.

Syntax

```
(HistoryGroupFlushTimes group_pattern days hours minutes)
```

Description

This command has not yet been documented.

HistoryGroupRemove

`HistoryGroupRemove` — is new, not yet documented.

Syntax

```
(HistoryGroupRemove group_pattern)
```

Description

This command has not yet been documented.

HistoryGroupStorageTimes

`HistoryGroupStorageTimes` — is new, not yet documented.

Syntax

```
(HistoryGroupStorageTimes group_pattern days hours minutes)
```

Description

This command has not yet been documented.

ignore

`ignore` — ignores a given point.

Syntax

`(ignore` *`pointname`*`)`

Arguments

`pointname`

The name of the point to be ignored.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command tells the VINE to ignore changes in value to the point *`pointname`*.

ignore_old_data

`ignore_old_data` — ignores changes with an old timestamp.

Syntax

```
(ignore_old_data 0|1)
```

Arguments

`0|1`

Use `1` to ignore old data, or `0` to not ignore old data.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command instructs the VINE to ignore any changes to data that arrive with an old timestamp. This corresponds to the Do not transmit changes with an old timestamp checkbox in the General option of the Properties window.

include

`include` — includes a file in with configuration files.

Syntax

```
(include filename [enabled] [as-sender-0|1])
```

Arguments

`filename`

The name of the configuration file to include.

`enabled`

One of the following flags indicating the enabled state.

- `1` means enabled; it will be used immediately after loading.
- `0` means disabled; it will be loaded but not used.

`as-sender-0|1`

Not yet documented.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command includes a file with configuration files. It can be used in a configuration file itself, causing the included file to be inserted into the configuration file at the point of this command. Or it can be sent to the VINE by another program. The *`enabled`* parameter puts the file into the enabled state, meaning that it is loaded and then immediately used. With that parameter turned off, the file is simply added to the VINE's list of configuration files.

instance

`instance` — creates an instance of a data organization model.

Syntax

```
(instance domain pointname assemblyname)
```

Arguments

`domain`

The domain of the model.

`pointname`

A name for the instance.

`assemblyname`

The assembly associated with the model.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates an instance of a data organization model, based on a given domain and assembly. For more information and an example, please refer to Section 13.4, *Data Organization*.

load_config_files

`load_config_files` — loads configuration files.

Syntax

```
(load_config_files 0|1)
```

Arguments

`0|1`

Use `1` to have configuration files loaded, or `0` to not have them loaded.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the VINE to load or not load its configuration files.

load_plugin

`load_plugin` — loads a specified plugin. (*experimental*)

Syntax

```
(load_plugin plugin_name run_now)
```

Arguments

`plugin_name`

The name of a plugin.

`run_now`

One of the following flags:

- `1` means the plugin will run immediately after it is loaded.
- `0` means disabled; the plugin will be loaded but not run.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the VINE to load a plugin.

The commands related to plugins are currently experimental.

load_scripts

`load_scripts` — loads scripts.

Syntax

```
(load_scripts filename enabled)
```

Arguments

`filename`

The name of a script.

`enabled`

One of the following flags indicating the enabled state.

- `1` means enabled; the script will be run immediately after it is loaded.
- `0` means disabled; the script will be loaded but not run.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the VINE to load or not load scripts.

lock

`lock` — locks and unlocks points.

Syntax

```
(lock name secur 0|1)
```

Arguments

`name`

The point name.

`secur`

The security level of the point.

`0|1`

Use `1` to lock the point, or `0` to unlock the point.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command locks and unlocks points. When a point is locked, it cannot be changed until it is unlocked.

log_file

log_file — sets up a log file.

Syntax

```
(log_file filename)
```

Arguments

logfile

The name of the log file.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets up a log file for messages that appear in the Event Log. If a log file already exists, this command changes that file's name. Starting and stopping logging is done with the **log_to_file** command. If logging to a file is enabled, the log messages will also continue to appear in the Event Log.

log_file_max

`log_file_max` — is new, not yet documented.

Syntax

```
(log_file_max kbytes)
```

Description

This command has not yet been documented.

log_to_file

`log_to_file` — starts or stops logging to a file.

Syntax

```
(log_to_file 0|1)
```

Arguments

Use `1` to start logging to a file, or `0` to stop.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command starts or stops logging error messages to the log file set up by **log_file**.

mirror_master

`mirror_master` — sets up a tunnelling (mirroring) master.

Syntax

```
(mirror_master host service localdomain [masterdomain])
```

Arguments

`host`

The master host's name or IP address.

`service`

The service name or port number of the mirroring master.

`localdomain`

The domain of the slave (i.e. local) computer

`masterdomain`

The domain of the mirroring master.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is to be sent to the slave VINE in a mirroring relationship. It tells the slave all the information it needs about its connection to the master VINE. This command corresponds to the entries in the Tunnel Master dialog that opens when you click the Add button in the Tunnel/Mirror option of the Properties window.

The VINE uses mirroring to do tunnelling, so the term 'mirror' is used to name the command.

mirror_master_2

`mirror_master_2` — sets up a secure tunnelling (mirroring) master.

Syntax

`(mirror_master_2 host port flagslocaldomain remotedomain heartbeat timeout [username password])`

Arguments

`host`

The master host's name or IP address.

`port`

The port number or service name of the mirroring master.

`flags`

Any combination of:

Hex	Flag	Notes
`0x00100000`	`READABLE`	Data is readable from the master. Always set this.
`0x00200000`	`WRITABLE`	Data is writable to the master.
`0x01000000`	`SYNC_SEND`	On initial connection, send all data to the master
`0x02000000`	`SYNC_RECV`	On initial connection, receive all data from the master
`0x04000000`	`SYNC_TIME`	On initial connection, synchronize by comparing time stamps.
`0x10000000`	`AUTHORITATIVE`	Master is authoritative. Data here is marked Not Connected when the connection drops.
`0x20000000`	`NON_AUTHORITATIVE`	I am authoritative. Data in the master is marked Not Connected when the connection drops.
`0x40000000`	`OVERRIDE_TIME`	Override the time stamp on incoming data with the system clock time.

Some combinations of flags will generate strange results:

- Combining `WRITABLE` with `OVERRIDE_TIME` will result in pollution of the master's time stamps.
- Only one of `SYNC_SEND`, `SYNC_RECV` and `SYNC_TIME` should be specified.
- Only one of `AUTHORITATIVE` and `NON_AUTHORITATIVE` should be specified.
- If `WRITABLE` is not set, then `SYNC_SEND`, `SYNC_TIME` and `NON_AUTHORITATIVE` make no sense.

`localdomain`

The domain of the slave (i.e. local) computer

`remotedomain`

The domain of the mirroring master.

heartbeat

The number of milliseconds for the heartbeat on this connection. `0` means disabled.

timeout

The number of milliseconds for the timeout on this connection. `0` means disabled. The timeout should be significantly longer than the heartbeat.

username

(optional) The user name to use if authenticating this connection.

password

(optional) The password to use if authenticating this connection. This password is stored in obfuscated text to stop somebody from casually copying it. However, since this obfuscation must be reversible to give access to the original password, it does not represent strong security. Do not allow untrusted people access to your configuration file.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is the same as **mirror_master**, but includes the necessary parameters for security. This command is sent to the slave VINE in a mirroring relationship. It tells the slave all the information it needs about its connection to the master VINE. This command corresponds to all the entries in the Tunnel Master dialog that opens when you click the Add button in the Tunnel/Nirror option of the Properties window.

The VINE uses mirroring to do tunnelling, so the term ’mirror’ is used to name the command.

mult

`mult` — multiplies the value of a point.

Syntax

```
(mult name number secs nano)
```

Arguments

`name`

The name of a point, which must be a number type.

`number`

A value to multiply by the value of the point.

`secs`

The time in seconds. If this is not specified, the current date and time in seconds is used.

`nano`

A fraction of a second, in nanoseconds. If this is not specified, the number of nanoseconds past the current date and time is used.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to multiply the value of a point.

private_attribute

`private_attribute` — creates a private attribute.

Syntax

```
(private_attribute assemblyname attrname type rw dflt_value [dflt_conf])
```

Arguments

`domain`

The domain to which this property applies.

`assemblyname`

The name of the assembly to which this attribute applies.

`attrname`

The name of the attribute.

`type`

A type for the private attribute.

`rw`

One of:

- `r` for read-only.
- `w` for write-only.
- `rw` for read-write.

`dflt_value`

A default value.

`dflt_conf`

A default confidence level. If nothing is entered, the system assumes `0`.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a private attribute. For information on the difference between an attribute and a private attribute, please refer to Section 13.4.3, *Attributes and Types*.

property

`property` — creates a property for an assembly.

Syntax

```
(property domain attrname propid propname type rw dflt_value [dflt_conf])
```

Arguments

`domain`

The domain to which this property applies.

`attrname`

The name of the attribute to which this property applies.

`propid`

An ID number, or `AUTO` to have the VINE assign an ID automatically.

`propname`

A name for the property.

`type`

A type for the property.

`rw`

One of:

- `r` for read-only.
- `w` for write-only.
- `rw` for read-write.

`dflt_value`

A default value.

`dflt_conf`

A default confidence level. If nothing is entered, the system assumes `0`.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a property. For more information and an example, please refer to Section 13.4.2, *Assemblies, Subassemblies, Attributes, and Properties*.

quality

`quality` — assigns a quality to a point.

Syntax

```
(quality name new_quality)
```

Arguments

`name`

The name of a point, as a string.

`new_quality`

The quality to be assigned to the point, as a number. Quality numbers are:

```
OPC_QUALITY_BAD = 0
OPC_QUALITY_UNCERTAIN = 0x40
OPC_QUALITY_GOOD = 0xc0
OPC_QUALITY_CONFIG_ERROR = 0x4
OPC_QUALITY_NOT_CONNECTED = 0x8
OPC_QUALITY_DEVICE_FAILURE = 0xc
OPC_QUALITY_SENSOR_FAILURE = 0x10
OPC_QUALITY_LAST_KNOWN = 0x14
OPC_QUALITY_COMM_FAILURE = 0x18
OPC_QUALITY_OUT_OF_SERVICE = 0x1c
OPC_QUALITY_WAITING_FOR_INITIAL_DATA = 0x20
OPC_QUALITY_LAST_USABLE = 0x44
OPC_QUALITY_SENSOR_CAL = 0x50
OPC_QUALITY_EGU_EXCEEDED = 0x54
OPC_QUALITY_SUB_NORMAL = 0x58
OPC_QUALITY_LOCAL_OVERRIDE = 0xd8
```

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command assigns a quality to a point. Typical qualities include `Good` and `Bad`.

read

`read` — reads a complete point definition.

Syntax

`(read` *`name`*`)`

Arguments

`name`

The name of a point.

Returns

The complete point definition in a message, with this syntax:

`(point` *`name type value`* `[`*`conf security locked seconds nanoseconds flags quality`*`])`

where:

`name`

The name of the point.

`type`

The data type of the point, one of integer, floating point, or character string.

`value`

The value of the point.

`conf`

The confidence level of the point, `0` - `100` percent, unused by most applications.

`security`

The security level of the point, `0` to `32768`, where higher numbers represent higher security.

`locked`

`0` for locked, or `1` for unlocked.

`seconds`

The operating system time in seconds when the point was read.

`nanoseconds`

The number of nanoseconds after *`seconds`* when the point was read.

`flags`

User-defined flags.

`quality`

A constant representing a quality of the connection, assigned by the VINE for this point, such as `Good`, `Bad`, `Last known`, `Local override`, etc. The possible values are those supported by OPC in Microsoft Windows.

Description

This command reads the current value of a point, along with all the other information it contains. See also **cread**.

report

`report` — requests notification of changes to a data point.

Syntax

```
(report name)
```

Arguments

`name`

The name of a data point.

Returns

A message with the complete definition of the point.

Description

This command requests the VINE to report changes (also called exceptions) to the value or any other information about a point, as soon as any change takes place. See also **creport**.

report_domain

`report_domain` — registers points and requests information on a whole domain.

Syntax

```
(report_domain domain flags)
```

Arguments

`domain`

The desired domain.

`flags`

Any bitwise-or combination of:

`DH_REG_ALL`	Register all points on this domain.
`DH_REG_FUTURE`	Register any new points created on this domain subsequent to this call.
`DH_REG_QUALIFY`	Tell the Cascade DataHub to emit all point names with the domain prepended, as in *`domain`*`:`*`point_name`*.
`DH_REG_ONCEONLY`	Tell the Cascade DataHub to send each point value exactly once.
`DH_REG_MODEL`	Tell the Cascade DataHub to send the data model as well as the point values.

Returns

One or more messages, depending on the *`flag`*(s) chosen. Each message contains the complete definition of the point.

Description

This command lets TCP client connections decide on a per-domain basis whether to informed of the data model for the domain, and whether to get future updates, fully-qualified domain names, or new points.

report_errors

`report_errors` — controls the reporting of errors.

Syntax

```
(report_errors 0|1 [error_point])
```

Arguments

`0|1`

If `0` (the default), errors in transmitting an exception will not be reported to the client. If `1`, an error string will be generated.

`error_point`

(Optional) The name of a point to contain the error string. If no name is given here, the error string will be written as the value of the point in which the error occurred.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command allows a client to specify how a failure to transmit a point change should be reported, either by modifying the point data and trying again, or by emitting an "error" point with the message in its data. The default is no reporting at all. In any case, if the attempt to emit the exception with the error information also fails, no further action is attempted by the Cascade DataHub.

request_initial_data

`request_initial_data` — gets current data when client connection is made.

Syntax

```
(request_initial_data yes|no|0|1)
```

Arguments

`yes|no|0|1`

Choose `yes` or `1` to request the data, or `no` or `0` to not make the request.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command causes the Cascade DataHub to send the current value of all points when the client connects.

save_config

`save_config` — forces the VINE to save its configuration.

Syntax

```
(save_config)
```

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command allows an external client to force the VINE to save its configuration.

secure

`secure` — adjusts the security level of a point.

Syntax

```
(secure name my_sec new_sec)
```

Arguments

`name`

The name of a VINE point, as a string.

`my_sec`

The user's security level.

`new_sec`

The new security level to be assigned to the point.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command adjusts the security level of a VINE point. If a point has a non-zero security level then any attempt to change the point value will fail if the writer claims to have a security level lower than the point's security level. The term "security level" is something of a misnomer because it is up to the writer to state what security level it has, and the writer could claim to have any security level. There is no validation of the claim by the VINE. Consequently, this security level is co-operative. It acts to stop errors among trusted programs, but does not act to limit access by untrusted programs.

The default security level for a point is 0.

set

`set` — sets the value of a point.

Syntax

```
(set name value [confidence])
```

Arguments

`name`

The name of the point. This is a string.

`value`

A string representation of the value for the point. The point value will be interpreted as integer, float or string based on the contents of the value string. This function tries each type in order, and uses the first type for which the value parameter is a valid representation. Double quotes around the value parameter are ignored. For example:

- `123` is an integer.
- `123.4` is a float.
- `1.234e2` is a float.
- `"123"` is an integer.
- `123abc` is a string.

`confidence`

A confidence factor in the range of `0` to `100` (optional). This is not used by the VINE, so is available to programs that produce graduated confidence, such as expert systems. If this value is not specified, it is set to `100`.

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (`\`) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with `\n`, `\r`, `\f`, `\t` respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets the value of a point. When this value is set, the following attributes of the point are set as follows:

- `seconds` and `nanoseconds` are set to the current time on the machine running the DataHub.
- `locked`, `sec`, and `quality` are all maintained at their previous values for this point.
- `flags` is set to `0`.

Please refer to the **write** command for more information about these parameters. See also **cset**, **force**, and **cforce**.

set_access

`set_access` — is new, not yet documented.

Syntax

```
(set_access pointname rw)
```

Description

This command has not yet been documented.

set_authoritative

`set_authoritative` — sets the type of a point.

Syntax

```
(set_authoritative domain flag)
```

Arguments

`domain`

The name of the data domain.

`flag`

A value of `1` indicates that this application is authoritative. A value of `0` indicates that VINE should be authoritative .

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command tells VINE that this application should be considered the authoritative source of data for this data domain. VINE will keep track of which data points this application has written to VINE, and will automatically mark those points as being `Not Connected` quality when the application disconnects. This provides a mechanism for VINE to provide quality information even if the source application exits unexpectedly.

There should only ever be one authoritative source of data for any data point, though VINE will not enforce this. If more than one application claims to be authoritative for a data point, that point will be given a `Not Connected` quality even if another authoritative application is still connected.

The command `(set_authoritative domain 0)` does NOT mean "turn off authoritative status". It informs VINE that the application is explicitly non-authoritative, and by extension that VINE should behave as if it is authoritative. Normally this is only meaningful between VINE instances constructing a tunnel. An application should not send `(set_authoritative domain 0)` to the DataHub. If an application does not want to be authoritative for a data domain, it simply should not send a **set_authoritative** command.

set_canonical

`set_canonical` — sets the type of a point.

Syntax

```
(set_canonical pointname canonical_type [force])
```

Arguments

`pointname`

The full name of the point, with domain.

`canonical_type`

Either a number with a legal numeric `VT_TYPE` value, or one of: `I1`, `I2`, `I4`, `UI1`, `UI2`, `UI4`, `CY`, `DATE`, `BOOL`, `BSTR`, `R4`, `R8`, `EMPTY`, `I1 ARRAY`, `I2 ARRAY`, `I4 ARRAY`, `UI1 ARRAY`, `UI2 ARRAY`, `UI4 ARRAY`, `CY ARRAY`, `DATE ARRAY`, `BOOL ARRAY`, `BSTR ARRAY`, `R4 ARRAY`, `R8 ARRAY`.

`force`

A value of `1` forces a change in canonical type. A value of `0`, or the omission of this optional parameter, will allow the canonical type to change only if it is currently `EMPTY`.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command sets the canonical type of a point. Normally a point's canonical type is `EMPTY`, meaning that it will maintain the data type of whatever data is written to it. If the canonical type is other than `EMPTY`, then any data written to the point will be converted to that type before it is stored.

When a point has a non-empty canonical type it is possible that the conversion could fail, in which case the point value is not changed.

show_data

`show_data` — displays the Data Browser.

Syntax

```
(show_data 0|1)
```

Arguments

`0|1`

Use `1` to show the Data Browser, or `0` to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command shows or hides the Data Browser.

show_debug_messages

show_debug_messages — show or hide debugging messages in the Data Browser.

Syntax

```
(show_debug_messages 0|1)
```

Arguments

`0|1`

Use `1` to show debugging messages in the Data Browser, or `0` to not show them.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you toggle the functionality of the Debug button in the Event Log on or off programmatically. The Debug option is very verbose, and could put a high demand on system resources.

show_event_log

`show_event_log` — displays the Event Log.

Syntax

```
(show_event_log 0|1)
```

Arguments

`0|1`

Use `1` to show the Event Log, or `0` to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command shows or hides the Event Log.

show_icon

`show_icon` — displays the system tray icon.

Syntax

```
(show_icon 0|1)
```

Arguments

`0|1`

Use `1` to display the system tray icon, or `0` to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you show or hide the system try icon ( ).

show_properties

`show_properties` — displays the Properties window.

Syntax

```
(show_properties 0|1)
```

Arguments

`0|1`

Use `1` to display the Properties window, or `0` to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command shows or hides the Properties window.

show_script_log

`show_script_log` — displays the Script Log.

Syntax

```
(show_script_log 0|1)
```

Arguments

`0|1`

Use `1` to display the Script Log, or `0` to hide it.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command shows or hides the Script Log.

subassembly

`subassembly` — creates a subassembly.

Syntax

```
(subassembly domain assemblyname subassemblyname instancename)
```

Arguments

`domain`

The name of the domain in which this subassembly will be created.

`assemblyname`

The name of the parent assembly.

`subassemblyname`

The name for this subassembly.

`instancename`

The instance name for this instance of the subassembly.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a subassembly level of a data organization. Unlike assemblies, each subassembly is instantiated when it is created, and therefore needs an instance name. For more information about assemblies and subassemblies with an example, please refer to Section 13.4.2, *Assemblies, Subassemblies, Attributes, and Properties*.

success

`success` — sets up a `success` message.

Syntax

```
(success command resultstring)
```

Arguments

`command`

The name of a command for which this success string will be used.

`resultstring`

A message string to be send to the issuer of a command when the command executes successfully. Also see Return Syntax.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is used to create messages that are sent from the VINE to participating programs whenever a given *`command`* is successfully executed.

tcp_service

`tcp_service` — sets a TCP service name or port number for incoming slave connections.

Syntax

```
(tcp_service service|port)
```

Arguments

`service|port`

The TCP service name or port number for incoming slave connections.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command informs a VINE acting as a master which TCP service name or port number it should listen on for incoming slave connections.

timeout

`timeout` — suspends data flow.

Syntax

```
(timeout ms)
```

Arguments

`ms`

The number of milliseconds to pause.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command causes the Cascade DataHub to stop sending data to the client for *`ms`* milliseconds.

TunnelEnable

`TunnelEnable` — is new, not yet documented.

Syntax

```
(TunnelEnable 0|1 host_pattern [remote_domain_pattern])
```

Description

This command has not yet been documented.

TunnelEnablePlain

`TunnelEnablePlain` — is new, not yet documented.

Syntax

```
(TunnelEnablePlain 0|1)
```

Description

This command has not yet been documented.

TunnelEnableSlave

`TunnelEnableSlave` — is new, not yet documented.

Syntax

```
(TunnelEnableSlave 0|1)
```

Description

This command has not yet been documented.

TunnelEnableSSL

`TunnelEnableSSL` — is new, not yet documented.

Syntax

```
(TunnelEnableSSL 0|1)
```

Description

This command has not yet been documented.

TunnelPlainPort

`TunnelPlainPort` — is new, not yet documented.

Syntax

```
(TunnelPlainPort port)
```

Description

This command has not yet been documented.

TunnelSlaveStatus

`TunnelSlaveStatus` — is new, not yet documented.

Syntax

```
(TunnelSlaveStatus host_pattern)
```

Description

This command has not yet been documented.

TunnelSSLCert

`TunnelSSLCert` — is new, not yet documented.

Syntax

```
(TunnelSSLCert filename)
```

Description

This command has not yet been documented.

TunnelSSLPort

`TunnelSSLPort` — is new, not yet documented.

Syntax

```
(TunnelSSLPort port)
```

Description

This command has not yet been documented.

transmit_insignificant

`transmit_insignificant` — permits transmission of insignificant changes.

Syntax

```
(transmit_insignificant 0|1)
```

Arguments

`0|1`

Use `1` to transmit insignificant changes, or `0` to not transmit them.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command permits transmission of insignificant changes. A change is considered insignificant if a point change differs from the existing value only by its `timestamp`. A change is considered significant if any attribute other than `timestamp` changes. That would include `lock`, `security`, `confidence`, `quality`, `value`. This command does not put any kind of deadband on the value.

type

`type` — creates a type.

Syntax

```
(type domain attrname [superattrname])
```

Arguments

`domain`

The domain in which this type applies.

`attrname`

The name of this type, which is the same as the name of the attribute.

`superattrname`

An attribute from which this type is derived.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command creates a type. For more information and an example, please refer to Section 13.4.3, *Attributes and Types*.

unload_plugin

`unload_plugin` — unloads a plugin. (*experimental*)

Syntax

```
(unload_plugin plugin_name)
```

Arguments

`plugin_name`

The name of a plugin.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you instruct the VINE to unload a plugin.

The commands related to plugins are currently experimental.

unreport

`unreport` — allows a client to stop receiving data value changes to a point.

Syntax

```
(unreport name)
```

Arguments

`name`

The name of the point to stop receiving values for.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command allows a client to stop future changes from being sent on a specified point.

unreport_domain

`unreport_domain` — is new, not yet documented.

Syntax

```
(unreport_domain domain)
```

Description

This command has not yet been documented.

version

`version` — returns the current version number.

Syntax

```
(version)
```

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command gives the current version number of the VINE.

write

`write` — writes information to a point.

Syntax

```
(write name type value conf sec locked seconds nanoseconds [flags quality])
```

Arguments

`name`

The name of the point. This is a string.

`type`

The type of point. One of

- `0` = string
- `1` = float (8-byte double)
- `2` = integer (32-bit integer)

`value`

A string representation of the value for the point. It will be interpreted into the type specified by the *`type`* parameter.

`conf`

A confidence factor in the range of `0` to `100`. This is not used by the VINE, so is available to programs that produce graduated confidence, such as expert systems.

`sec`

A security level for this point. This is rarely used. If a point's security level is set to a non-zero value then attempts to write to that point must claim a security level equal to or greater than the security level of the point. This uses a "good citizen" model - the writer can claim any security it wants, and is assumed to be honest - so there is no strong security here. It is intended for systems that want to avoid accidental changes to values. Security level can be from `0` to `32767`.

`locked`

An indication that the point is locked, and cannot be changed. Can be `0` or `1`. Attempts to write to a locked point will fail.

`seconds`

The UNIX epoch - seconds since Jan. 1, 1970, as produced by the `time()` function.

`nanoseconds`

The number of nanoseconds inside this second. Cannot exceed `1,000,000,000`.

`flags`

User level code should always send a `0` for this value.

`quality`

A quality indicator consistent with the OPC DA specification. This is not a bit field. It can be one of:

`PT_QUALITY_BAD`	`0`
`PT_QUALITY_UNCERTAIN`	`0x40`
`PT_QUALITY_GOOD`	`0xc0`

```
PT_QUALITY_CONFIG_ERROR             0x04
PT_QUALITY_NOT_CONNECTED            0x08
PT_QUALITY_DEVICE_FAILURE           0x0c
PT_QUALITY_SENSOR_FAILURE           0x10
PT_QUALITY_LAST_KNOWN               0x14
PT_QUALITY_COMM_FAILURE             0x18
PT_QUALITY_OUT_OF_SERVICE           0x1c
PT_QUALITY_WAITING_FOR_INITIAL_DATA 0x20

PT_QUALITY_LAST_USABLE              0x44
PT_QUALITY_SENSOR_CAL               0x50
PT_QUALITY_EGU_EXCEEDED             0x54
PT_QUALITY_SUB_NORMAL               0x58
PT_QUALITY_LOCAL_OVERRIDE           0xd8
```

All strings can be surrounded by double-quotes if the string contains spaces or special characters. The backslash character (\) escapes double quotes and backslashes within the string. Newline, carriage return, form feed and tab are represented with `\n`, `\r`, `\f`, `\t` respectively. Strings must not contain newline characters.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command lets you manually write information to a point. See also **cwrite**.

II. Obsolete and Unused Commands

Table of Contents

asyncsocket 366
authgroup 367
authuser 368
bandwidth_reduce 369
deleted 370
drop_license 371
echo 372
enable_connect_server 373
EnableDDEServer 374
exception_buffer 375
failed_license 376
flush 377
flush_log 378
master_host 379
master_service 380
on_change 381
point 382
qnx_name_attach 383
qnx_receiver 384
readid 385
register_datahub 386
report_all 387
report_datahubs 388
request 389
run 390
script_register 391
script_symbol 392
slave 393
sync 394
taskdied 395
taskstarted 396
using_license 397
warn_of_license_expiry 398

This reference contains commands that are deprecated, obsolete, and no longer used, as well as commands that are for internal use only. These commands should rarely if ever be used.

asyncsocket

`asyncsocket` — sets up asynchronous communication on a socket.

Syntax

```
(asyncsocket socket)
```

Arguments

`socket`

The socket address, as a string.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command tells the VINE to communicate asynchronously on the specified *`socket`*.

authgroup

`authgroup` — is deprecated.

Syntax

```
(authgroup name bits max_concurrent_logins max_logins expiry)
```

authuser

`authuser` — is deprecated.

Syntax

```
(authuser name group hash bits max_concurrent_logins max_logins login_count expiry)
```

bandwidth_reduce

`bandwidth_reduce` — is for internal use only.

deleted

`deleted` — checks if a point has been deleted.

Syntax

```
(deleted pointname [domain])
```

Arguments

`pointname`

A string containing the name of the point.

`domain`

The domain of the point. If not specified, the `default` domain is used.

Returns

A message indicating whether the point has been deleted.

Description

This command checks if the given *`point`* has been deleted.

We do not recommend deleting points, as it could cause unexpected behavior for other users of the point.

drop_license

`drop_license` — is for internal use only.

echo

echo — is for internal use.

Syntax

```
(echo name type value [conf security locked seconds nanoseconds flags quality])
```

Description

This command is only used between two VINEs. A non-VINE client should never issue an **echo** command.

enable_connect_server

`enable_connect_server` — is deprecated.

Syntax

```
(enable_connect_server 0|1)
```

EnableDDEServer

`EnableDDEServer` — is for internal use only.

Syntax

```
(EnableDDEServer 0|1)
```

Description

This command is for internal use only. To enable the DDE server, please refer to **enable_dde_server**.

exception_buffer

`exception_buffer` — is deprecated.

Syntax

```
(exception_buffer bytes)
```

failed_license

`failed_license` — is for internal use only.

flush

`flush` — flushes output to a terminal (Linux).

Syntax

```
(flush)
```

Arguments

none

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is not used at all in Windows, and even in Linux and QNX there is really not much need for it. The command sends a flush to the terminal, so that if the VINE is printing to a terminal and has not completely flushed its output, this will cause all pending printed characters to appear.

flush_log

`flush_log` — is deprecated.

Syntax

```
(flush_log)
```

master_host

`master_host` — is deprecated in favor of **mirror_master**.

Syntax

```
(master_host name|IP)
```

master_service

`master_service` — is deprecated in favor of **mirror_master**.

Syntax

```
(master_service service|port)
```

on_change

on_change — is for internal use only.

point

`point` — is used internally.

Syntax

```
(point name type value [conf security locked seconds nanoseconds [quality]])
```

Arguments

`name`

The name of the point.

`type`

The type of the point.

`value`

The value of the point.

`conf`

The confidence level of the point.

`security`

The security level of the point.

`locked`

The locked status of the point.

`seconds`

The current time in seconds.

`nanoseconds`

The number of nanoseconds past *`seconds`*.

`quality`

The quality of the point.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is the string that the VINE sends to all clients when a point value changes (i.e. a point exception occurs). The DataHub also listens for a **point** command because this is the mechanism that it uses to get updates from another VINE that it is tunnelling/mirroring. This is not a command that a user would normally emit. Point changes should be sent to the VINE using the **write** or **cwrite** commands.

qnx_name_attach

`qnx_name_attach` — does nothing.

Syntax

`(qnx_name_attach` *`node name`*`)`

qnx_receiver

`qnx_receiver` — does nothing.

Syntax

`(qnx_receiver` *`name`*`)`

readid

`readid` — should *not* be used.

Syntax

`(readid` *`pointnumber`*`)`

Arguments

`pointnumber`

The nth point in the sender's `default` domain.

Returns

A message indicating success or error. Please refer to Return Syntax for details.

Description

This command is like read but not as useful or robust. It reads the nth point in the point table of the sender's `default` domain. This is neither useful nor not robust, since deleting a point will cause unpredictable behavior. Avoid using this command.

register_datahub

`register_datahub` — replaced by **report_domain**.

Syntax

```
(register_datahub domain)
```

report_all

`report_all` — replaced by **report_domain**.

Syntax

```
(report_all future domain_needed)
```

report_datahubs

`report_datahubs` — does nothing.

Syntax

```
(report_datahubs yes|no)
```

request

`request` — replaced by **report_domain**.

Syntax

`(request` *`pointname`*`)`

run

`run` — does nothing.

Syntax

```
(run command [argument...])
```

script_register

`script_register` — is for internal use only.

script_symbol

`script_symbol` — is for internal use only.

slave

`slave` — is for internal use only.

sync

sync — is for internal use only.

taskdied

`taskdied` — is for internal use.

Syntax

```
(taskdied name domn qu nodename node pid chid)
```

taskstarted

`taskstarted` — is for internal use.

Syntax

```
(taskstarted name domn qu nodename node pid chid)
```

using_license

`using_license` — is for internal use only.

warn_of_license_expiry

`warn_of_license_expiry` — is deprecated.

Syntax

```
(warn_of_license_expiry 0|1)
```

Index

A

acksuccess, 238
activation
  for bridging, 143
add, 239
  configuration file, 169
  data domain, 138
  DDE client, 147
  DDE server, 148
  Security client, 171
add a timestamp, 137
Aggregation
  and Bridging, ?
  and Tunnelling, ?
  configuring the DataHub, ?
  OPC, ?
alias, 240
alive, 241
append, 242
assembly, 243
asyncsocket, 366
attribute, 244
auth, 245
authentication, 114
authgroup, 367
authorization, 116
authreload, 246
authuser, 368
auto_create_domains, 247
auto_timestamp, 248

B

bandwidth_reduce, 369
bridge, 249
Bridges
  configuring, 38
bridge_remove, 251
bridge_remove_pattern, 252
bridge_transform, 253
bridging
  activation, 143
  configuration window, 143
  creating data sets, ?
  direction, 146
  local servers, ?
  OPC, 38
  point display, 146
  point selection, 144
  status, 143
  to Excel, ?
  transformations, 145
Bridging Properties, 142

C

cforce, 255
client
  OPC, ?
  OPC A&E, ?
Cogent DataHub
  exiting, 11
  installing, 1
  running, 11
  uninstalling, 1
COM
  security, ?
Computer Name
  OPC, ?
  OPC A&E, ?
configuration file
  add, remove, move, load, 169
configuration window
  bridging, 143
configure
  database, 152
  email, 162
  historian, 156
  quicktrend, 148
  redundancy, 164
  System Monitor, 160
  update, 169
  web server, ?
  webview, ?
configure database
  for data logging, 152
Configured actions
  database queries, 83
  email, 104
  logging, 68
Configuring
  bridges, 38
  the mail server, 92
Configuring a database table, 56
  for database queries, 74
Configuring the DataHub
  for aggregation, ?
connecting
  to InTouch, ?
  to Linux, ?

to QNX, ?
Connection Name
DDE, 147
OPC, ?
OPC A&E, ?
connections
Excel and Cogent DataHub, 19
CPU saturation
optimizing, 133
cread, 256
create, 257
create_domain, 258
creport, 259
cset, 260
cwrite, 261

D

data domain
add, 138
remove, 138
Data Domain Name
OPC, ?
OPC A&E, ?
data logging
configure database, 152
how to use, 48
store and forward, 153
data sets
creating to, ?
data throughput
optimizing, 131
database
concepts, 207
configure, 152
query, 155
terminology, 207
write to, 152
Database Properties, 152
Database queries
assigning a trigger, 77
configured actions, 83
configuring a database table, 74
how to, 69
quick start, 69
setting up DSN, 72
trigger conditions, 79
DataHub
scripting;, 109
security;, 112
DDE
non-English characters, 148
DDE client
add, remove, edit, enable, 147
DDE Item Definition Window, 147
DDE Properties, 146
DDE server
add, remove, enable, 148
DDEAdvise, 263
with drag and drop, 19
DDEConnect, 264
DDEDisconnect, 265
DDEExcelUnicode, 266
DDEInit, 267
DDERequest
in a macro, 21
DDEService, 268
DDEUnadvise, 269
DDEUnadvisePattern, 270
DDEUnadvisePoint, 271
debug, 272
defaultprop, 273
delete, 274
deleted, 370
direction
for bridging, 146
div, 275
domain, 276
domains, 277
domain_bridge, 278
domain_bridge_enable, 280
domain_bridge_prefer, 281
domain_bridge_refresh, 282
domain_bridge_remove, 283
drop_license, 371
DSN
setting up for logging, 54
setting up for queries, 72
dump, 284

E

echo, 372
edit
DDE client, 147
Security client, 171
Email
assigning a trigger, 99
configure, 162
configured actions, 104
defining a message, 97
how it works, 91
how to send, 91
HTML messsage examples, 104

test message, 93
trigger conditions, 101
Email Properties, 162
Email server
configuring, 92
EnableDDEServer, 374
enable_bridging, 285
enable_connect_server, 373
enable_dde_client, 286
enable_dde_server, 287
enable_domain_bridging, 288
enable_opc_client, ?
enable_opc_server, ?
enable_scripting, 289
enter
licenses, 176
error, 290
Excel
arrays, 27
bridging to, ?
connecting to the Cogent DataHub, 19
data from, 22
data into, 19
data to DDE Client, 23
macro to get data, 21
macro to send data, 25
networking, ?
ranges, 27
exception_buffer, 375
execute_plugin, 291
exit, 292
exiting the DataHub, 11

F

failed_license, 376
flush, 377
flush_log, 378
force, 293
format, 294

G

General Properties, 137
get_client_stats, 295

H

heartbeat, 296
help
pop-up, 11
historian
configure, 156
Historian Properties, 156
HistorianAdd, 297
HistorianFlags, 298
HistorianRemove, 299
HistorianSaveConfig, 300
HistorianSetConfiguring, 301
HistoryGroupAdd, 302
HistoryGroupAddPoint, 303
HistoryGroupDeadband, 304
HistoryGroupDefault, 305
HistoryGroupFileTimes, 306
HistoryGroupFlushTimes, 307
HistoryGroupRemove, 308
HistoryGroupStorageTimes, 309
How to
query a database, 69
How to send
email, 91
text messages, 91
How to use
data logging, 48
the mailer, 91
the System Monitor, 84
the web server, ?
HTML messsage examples
email, 104

I

ignore, 310
ignore_old_data, 311
include, 312
installing
updates, 7
installing the Cogent DataHub, 1
instance, 313
InTouch
connections, ?
Item Names
DDE, 147

K

Key columns
for logging, 60

L

licenses
  entering, 176
  loading, 175
  removing, 176
  string, 175
Licenses Properties, 175
Linux
  connections, ?
  to OPC, Linux setup, ?
  to OPC, Windows setup, ?
load
  configuration file, 169
load_config_files, 314
load_plugin, 315
load_scripts, 316
lock, 317
Logging
  assigning a trigger, 61
  configured actions, 68
  configuring a database table, 56
  key columns, 60
  quick start, 48
  setting up DSN, 54
  trigger conditions, 63
log_file, 318
log_file_max, 319
log_to_file, 320

M

macro to get data, 21
macro to send data, 25
Mail server
  configuring, 92
Mailer
  how to use, 91
master
  for tunnelling/mirroring, 142
master_host, 379
master_service, 380
message buffer
  maximum size, 137
mirror_master, 321
mirror_master_2, 322
move
  configuration file, 169
mult, 324

N

networking
  Excel, ?
  System Monitor, 87

O

ODBC, 207
old value queuing
  optimizing, 131
on_change, 381
OPC
  Aggregation, ?
  Bridging, ?
  client, ?
  Computer Name, ?
  Connection Name, ?
  Data Domain Name, ?
  server, ?
  Server Name, ?
  to Linux, Linux setup, ?
  to Linux, Windows setup, ?
  to QNX, QNX setup, ?
  to QNX, Windows setup, ?
  Tunnelling, ?
OPC A&E
  client, ?
  Computer Name, ?
  Connection Name, ?
  Data Domain Name, ?
  server, ?
  Server Name, ?
OPC DA Properties, ?
OPC Properties, ?
OPCActivate, ?
OPCAddItem, ?
OPCAddItem2, ?
OPCAEAttach, ?
OPCAEDetach, ?
OPCAEEnable, ?
OPCAEEnableClient, ?
OPCAEEnableServer, ?
OPCAEFilter, ?
OPCAEServerInit, ?
OPCApply, ?
OPCAttach, ?
OPCAttach2, ?
OPCConnect, ?
OPCDetach, ?
OPCEnable, ?
OPCEnableClient, ?
OPCEnableServer, ?

OPCInit, ?
OPCMinimumSecurity, ?
OPCModify, ?
OPCQueryConnection, ?
OPCQueryConnections, ?
OPCQueryPoint, ?
OPCQueryPointPattern, ?
OPCQueryPoints, ?
OPCRefresh, ?
OPCReload, ?
OPCRemoveItem, ?
optimizing
  CPU saturation, 133
  data throughput, 131
  old value queuing, 131
  screen output, 132
  slow networks, 131
  TCP connections, 131
  un-buffered delivery, 132
options
  Web Server, ?

P

passwords, 122
point, 382
point display
  for bridging, 146
point selection
  for bridging, 144
pop-up
  help, 11
private_attribute, 325
Properties
  Bridging, 142
  Database, 152
  DDE, 146
  Email, 162
  General, 137
  Historian, 156
  Licenses, 175
  OPC A&E, ?
  OPC DA, ?
  Quick Trend, 148
  Redundancy, 164
  Scripting, 169
  Security, 171
  System Monitor, 160
  Tunnelling/Mirroring, 139
  Update, 169
  Web Server, ?
  WebView, ?
property, 326

Q

QNX
  connections, ?
  to OPC, QNX setup, ?
  to OPC, Windows setup, ?
qnx_name_attach, 383
qnx_receiver, 384
quality, 327
query
  database, 155
Quick start
  database queries, 69
  logging, 48
quicktrend
  configure, 148
QuickTrend Properties, 148

R

read, 328
readid, 385
reading data, 123
redundancy
  configure, 164
Redundancy Properties, 164
register_datahub, 386
remove
  configuration file, 169
  data domain, 138
  DDE client, 147
  DDE server, 148
  licenses, 176
  Security client, 171
report, 329
report_all, 387
report_datahubs, 388
report_domain, 330
report_errors, 331
request, 389
request_initial_data, 332
run, 390
running the DataHub, 11

S

save_config, 333
screen output
  optimizing, 132
scripting
  DataHub, 109
Scripting Properties, 169
script_register, 391
script_symbol, 392
secure, 334
security
  authentication, 114
  authorization, 116
  COM, ?
  DataHub, 112
  passwords, 122
Security client
  add, remove, edit, enable, 171
Security Properties, 171
server
  OPC, ?
  OPC A&E, ?
Server Name
  OPC, ?
  OPC A&E, ?
Service
  DDE, 147
set, 335
set_access, 336
set_authoritative, 337
set_canonical, 338
show_data, 339
show_debug_messages, 340
show_event_log, 341
show_icon, 342
show_properties, 343
show_script_log, 344
slave, 393
  for tunnelling/mirroring, 139
slow networks
  optimizing, 131
SMS text messaging, 91
SQL, 208
status
  of bridging, 143
store
  transformations, 144
store and forward
  for data logging, 153
string
  license, 175
subassembly, 345
success, 346
sync, 394
System Monitor
  configure, 160
  how to use, 84
  networking, 87
System Monitor Properties, 160
system requirements, 1

T

taskdied, 395
taskstarted, 396
TCP connections
  optimizing, 131
tcp_service, 347
Text messages
  how to send, 91
Text messaging
  with SMS, 91
timeout, 348
Topic
  DDE, 147
transformations
  for bridging, 145
  storing, 144
transmit changes
  insignificant, 137
  old timestamp, 137
transmit_insignificant, 357
Trigger
  for database queries, 77
  for emails, 99
  for logging, 61
Trigger conditions
  database queries, 79
  email, 101
  logging, 63
TunnelEnable, 349
TunnelEnablePlain, 350
TunnelEnableSlave, 351
TunnelEnableSSL, 352
Tunnelling
  and Bridging, ?
  and Aggregation, ?
  connecting to an OPC client, 34
  connecting to an OPC server, 33
  OPC, ?
  part of the data set, ?
  testing the connection, ?
tunnelling/mirroring

master, 142
slave, 139
Tunnelling/Mirroring Properties, 139
TunnelPlainPort, 353
TunnelSlaveStatus, 354
TunnelSSLCert, 355
TunnelSSLPort, 356
type, 358

U

un-buffered delivery
optimizing, 132
uninstalling the DataHub, 1
unload_plugin, 359
unreport, 360
unreport_domain, 361
update
configure, 169
Update Properties, 169
updates
installiing, 7
using_license, 397

V

version, 362

W

warn_of_license_expiry, 398
web server
configure, ?
how to use, ?
options, ?
Web Server Properties, ?
webview
configure, ?
WebView Properties, ?
write, 363
write to
database, 152
writing data, 123

Colophon

This book was produced by Skkynet Cloud Systems, Inc. from a single-source group of SGML files. Gnu Emacs was used to edit the SGML files. The DocBook DTD and related DSSSL stylesheets were used to transform the SGML source into HTML, PDF, and QNX Helpviewer output formats. This processing was accomplished with the help of OpenJade, JadeTeX, Tex, and various scripts and makefiles. Details of the process are described in our book: Preparing Cogent Documentation, which is published on-line at `http://developers.cogentrts.com/cogent/prepdoc/book1.html`.

Text and illustrations by Bob McIlvride and Paul Benford.